IMPLEMENTASI MIKROPROSESOR 8085 DAN SISTEM MEMORI DALAM *FIELD PROGRAMMABLE GATE ARRAY* (FPGA)

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Teknik

DISUSUN OLEH:

HAFRIDA RAHMAH

NIM. 0710630021-63

KEMENTERIAN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2011

LEMBAR PERSETUJUAN

IMPLEMENTASI MIKROPROSESOR 8085 DAN SISTEM MEMORI DALAM *FIELD PROGRAMMABLE GATE ARRAY* (FPGA)

SKRIPSI

Diajukan Untuk Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik

Disusun oleh:
HAFRIDA RAHMAH
NIM. 07106300021-63

Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	**Dosen Pembimbing II**
Moch. Rif'an, ST., MT.	**Dr. Agung Darmawansyah, ST., MT.**
NIP. 19710301 200012 1 001	**NIP. 19721218 199903 1 002**

LEMBAR PENGESAHAN

IMPLEMENTASI MIKROPROSESOR 8085 DAN SISTEM MEMORI DALAM *FIELD PROGRAMMABLE GATE ARRAY* (FPGA)

Disusun oleh :

HAFRIDA RAHMAH

NIM. 0710630021-63

Skripsi ini telah diuji dan dinyatakan lulus
pada tanggal **23 Desember 2011**

Majelis Penguji

Ir. M. Julius St., MS.	**Ir. Ponco Siwindarto, M.Eng.Sc.**
NIP. 19540720 198203 1 002	**NIP. 19590304 198903 1 001**

Ir. Nurussa'adah, MT.
NIP. 19680706 199203 2 001

Mengetahui:

Ketua Jurusan Teknik Elektro

DR. Ir. Sholeh Hadi P., MS.
NIP. 19580728 198701 1 001

PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karuniaNya, penulis dapat menyelesaikan skripsi yang berjudul "Implementasi Mikroprosesor 8085 dan Sistem Memori Dalam *Field Programmable Gate Array (FPGA*)". Skripsi ini disusun sebagai persyaratan untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan bebagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada:

- Ibu, Ayah, dan Nenek atas segala nasihat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Kakak-kakak penulis yang banyak mendoakan hingga terselesaikannya skripsi ini,
- Bapak Dr. Ir. Sholeh Hadi Pramono, MS. selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak M. Aziz Muslim, ST., MT, Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak M. Julius St, Ir., MS selaku Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Moch. Rif'an, ST., MT. selaku Dosen Pembimbing I atas segala bimbingan, pengarahan, gagasan, ide, saran serta motivasi yang telah diberikan,
- Bapak Dr. Agung Darmawansyah, ST., MT selaku Dosen Pembimbing II atas segala bimbingan, pengarahan, saran, kritik, dan masukan yang telah diberikan,
- Staf Rekording, staf Pengajaran, dan staf Ruang Baca Jurusan Teknik Elektro yang telah membantu segala urusan penulis selama ini,
- Sahabat-sahabat penulis Fyta_pytung, Dela, Rizki_Kecil, Anno, Puput, Maul, dan Utari atas saran, semangat, pelajaran, pengalaman dan kebersamaan yang telah diberikan,
- *GBEmate* Yuli, Rizal, dan Anas atas waktu dan telinga kalian; *classmate* Doyot, Inas, dan Tika; Mas Kanzi, Mas Arif, Yudo, Aka_ct, Gladi, Pa'i, dan 2PM,
- Teman-teman Laboratorium SisDig, Workshop, pejuang PKM, dan Tim Roket Rintisan "Al-Fatih dan "E-Fly 63",

- Teman-teman CORE angkatan 2007 yang telah berbagi dalam hal duka maupun suka dengan penulis dan selalu mengajarkan arti kebersamaan,
- Seluruh teman-teman, senior serta semua pihak yang tidak mungkin untuk dicantumkan namanya satu-persatu, terima kasih banyak atas bantuan dan dukungannya.

Penulis menyadari bahwa tugas akhir ini masih belum sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis berharap, semoga tugas akhir ini bermanfaat bagi kita semua.

Malang, Desember 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Hafrida Rahmah, Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Desember 2011, *Implementasi Mikroprosesor 8085 dan Sistem Memori Dalam Field Programmable Gate Array (FPGA*), Dosen Pembimbing : Moch. Rif'an, ST., MT. dan Dr. Agung Darmawansyah ST., MT.

Mikroprosesor adalah sebagai pengontrol atau pengolah utama dalam suatu rangkaian elektronik. Kecenderungan terhadap penggunaan mikroprosesor bertahan cukup lama hingga dikembangkannya mikrokontroler. Namun, penggunaan mikroprosesor sebagai dasar pembelajaran masih tetap dikembangkan dan belum dapat tergantikan.Kebutuhan akan mikroprosesor untuk pembelajaran di institusi berbanding terbalik dengan ketersediaan mikroprosesor dan penunjangnya saat ini contohnya pada praktikum sistem digital di Laboratorium Sistem Digital TEUB. Oleh karena itu, diperlukan pengganti mikroprosesor 8085 yang tetap dapat berfungsi sebagaimana seperti mikroprosesor 8085 sehingga dirancanglah sistem pengganti mikroprosesor yang ditanamkan dalam FPGA.

Sistem pengganti ini juga harus mempunyai memori dan unit input output agar dapat langsung digunakan tanpa penambahan perangkat lain. Mikroprosesor 8085 ini harus dapat digunakan untuk memproses suatu instruksi berdasar set instruksi 8085 dengan sinyal kontrol yang dimilikinya,memiliki frekuensi clock sesuai rentang besaran yang dapat dimiliki, dapat membaca maupun menulis data ke unit luar yaitu unit input output dan memori. Hasil pengujian menunjukkan bahwa sistem mikroprosesor 8085 dapat memproses instuksi-instruksi yang diperintahkan sesuai dengan beberapa instruksi dalam set instruksi 8085, memiliki keluaran sinyal kontrol yang sesuai dan frekuensi bekerja sesuai dengan range maksimum frekuensi mikroprosesor 8085 yaitu 3,25MHz sehingga kinerja lebih cepat jika dibandingkan modulkit praktikum mikroprosesor 8085 yang ada yaitu 2 MHz dan juga dapat menggantikan modul praktikum yang telah ada.

Kata kunci: Mikroprosesor, Clock, Sinyal kontrol, Memori, FPGA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sejak awal perkembangannya hingga saat ini, teknologi digital menjadi salah satu teknologi yang berkembang begitu pesat. Hampir semua teknologi analog digantikan dengan teknologi digital. Perangkat digital seperti pemutar MP3 tersusun atas sistem yang kompleks yang biasa dikenal dengan *integrated circuit* (*IC*). Salah satu perkembangan IC yang dikenal secara umum untuk pembelajaran suatu teknologi digital yang bekerja dengan program adalah mikroprosesor.

Mikroprosesor adalah salah satu perkembangan teknologi digital. Mikroprosesor adalah sebuah chip (IC) yang bekerja dengan program. Fungsi mikroprosesor adalah sebagai pengontrol atau pengolah utama dalam suatu rangkaian elektronik. Mikroprosesor biasa disebut juga CPU (Central Processing Unit). Kecenderungan terhadap penggunaan mikroprosesor bertahan cukup lama hingga dikembangkannya mikrokontroler. Namun, penggunaan mikroprosesor sebagai dasar pembelajaran masih tetap dikembangkan dan belum dapat tergantikan.

Kebutuhan akan mikroprosesor untuk pembelajaran di institusi berbanding terbalik dengan ketersediaan mikroprosesor dan penunjangnya saat ini. Sebagai contoh, praktikum mata kuliah mikroprosesor yang menggunakan mikroprosesor 8085 yang digunakan di beberapa laboratorium institusi perguruan tinggi semakin berkurang, padahal ketersediaan mikroprosesor ini di pasaran sudah tidak ada. Oleh karena itu, diperlukan pengganti mikroprosesor 8085 yang tetap dapat berfungsi sebagaimana seperti mikroprosesor 8085.

Sistem pengganti mikroprosesor 8085 jika diciptakan dengan proses produksi *IC* (*integrated circuit*) akan sangat kompleks, membutuhkan biaya, dan membutuhkan waktu pengerjaan yang lama sehinga hampir tidak memungkin bagi kita untuk membuatnya. Namun, saat ini telah ditemukan teknologi *PLD* (*programmable logic device*) yang salah satu contohnya adalah FPGA yaitu sebuah teknologi integrasi yang mampu diprogram menjadi suatu rangkaian sesuai dengan yang kita inginkan. Dengan adanya teknologi PLD maka sistem pengganti mikroprosesor 8085 dapat dirancang hingga berfungsi sebagaimana seperti mikroprosesor 8085.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan sebelumnya, dapat dirumuskan permasalahan sebagai berikut:

1) Bagaimana arsitektur mikroprosesor 8085 yang akan dirancang dalam FPGA sebagai media implementasi sistem.
2) Bagaimana sistem memori yang dapat kompatibel dengan mikroprosesor 8085.
3) Bagaimana rancangan sistem penunjang yang dibutuhkan untuk menghubungkan mikroprosesor dengan memori dan unit luar mikroprosesor.
4) Apakah sistem dapat digunakan sesuai dengan penggunaan mikroprosesor 8085 meliputi beberapa instruksi dalam *instruction set* mikroprosesor 8085.

1.3 Ruang Lingkup

Mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan rancangan akan diberi batasan sebagai berikut:

1) Membahas bagaimana sistem mikroprosesor 8085 bekerja.
2) Membahas bagaimana sistem memori dapat digunakan dengan mikroprosesor 8085.
3) Intruksi yang dapat dijalankan atau dieksekusi menggunakan instruksi-instruksi yang sering digunakan dalam praktikum mikroprosesor 8085.
4) Tidak membahas cara penulisan bahasa hardware yang akan digunakan pada sistem mikroprosesor 8085.

1.4 Tujuan

Penelitian ini bertujuan untuk merancang pengganti mikroprosesor 8085 yang sesuai dengan arsitektur mikroprosesor 8085 dan sistem memori sebagai pengganti alat praktikum dasar teknologi digital sehingga pembelajaran tentang teknologi digital dapat terus berlangsung.

1.5 Sistematika Penulisan

Skripsi ini terdiri dari enam bab dengan sistematika pembahasan sebagai berikut:

BAB I Pendahuluan

Membahas latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perancangan dan pembuatan alat, yang meliputi : arsitektur mikroprosesor 8085, arsitektur memori, perangkat modul praktikum mikroprosesor 8085 dan spesifikasinya, Field Programmable Gate Array.

BAB III Metodologi Penulisan

Membahas metode penelitian dan perencanaan sistem.

BAB IV Perancangan dan Pembuatan Alat

Membahas perancangan sistem mikroprosesor, jalannya sistem mikroprosesor beserta koordinasi mikroprosesor dengan unit-unit tambahan, serta perancangan tiap blok unit-unit penyusun sistem mikroprosesor, sistem bidirectional buffer dekoder, memori, serta input output.

BAB V Pengujian dan Analisis

Membahas hasil pengujian sistem untuk tiap-tiap blok dan secara keseluruhan.

Bab VI Kesimpulan dan Saran

Membahas kesimpulan perancangan ini dan saran-saran yang diperlukan untuk pengembangan selanjutnya.

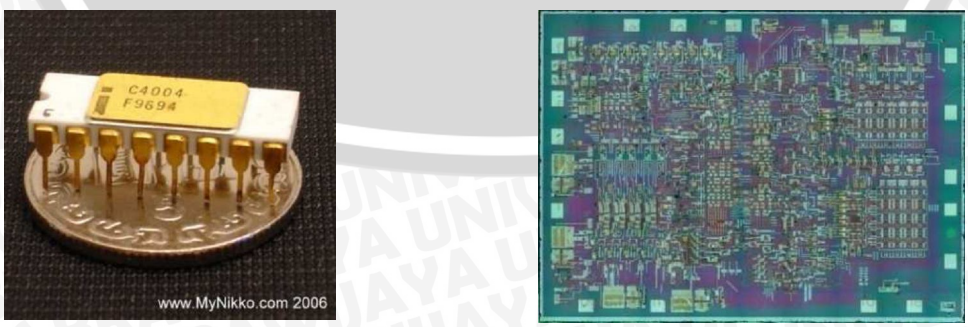
BAB II
TINJAUAN PUSTAKA

Bagian yang akan dibahas dalam sistem mikroprosesor 8085 dan memori ini adalah gambaran umum tentang mikroprosesor, mikroprosesor 8085, karakteristik memori yang dapat diakses oleh mikroprosesor 8085, serta FPGA sebagai devais perancangan program sistem mikroprosesor 8085.

2.1 Mikroprosesor

Mikroprosesor adalah seluruh atau sebagian besar representasi fungsi dari *central processing unit* (CPU) yang merupakan sebuah "mesin" pemproses yang dibuat dengan menggunakan teknologi mikro dan terintegrasi dalam sebuah *integrated circuit* (IC). Mikroprosessor didesain untuk melakukan operasi-operasi aritmatika dan logika. Operasi-operasi ini dilakukan di dalam mikroprosessor dengan bantuan beberapa blok penyimpan data yang disebut dengan *register*. Mikroprosessor pada umumnya dapat melakukan operasi penjumlahan, pengurangan, perbandingan dua angka, dan mengatur perpindahan suatu angka dari suatu tempat ke tempat yang lain. Operasi-operasi tersebut adalah hasil dari sebuah *set of instruction* yang juga merupakan bagian dari mikroprosessor.

Mikroprosessor yang pertama kali dibuat adalah mikroprosessor 4004 yang dibuat oleh Intel pada awal tahun 1970an. Mikroprosessor ini digunakan untuk kalkulator elektronik yang masin menggunakan 4-bit aritmatika BCD (*binary code decimal*). Dengan adanya *general-purpose microprocessor* pertama ini, perkembangan teknologi di bidang pemprosessan data menjadi sangat pesat. Pembuatan-pembuatan mikroprosessor 8 hingga 16 bit-pun mengawali kemajuan mikrokomputer di pertengahan era 70an. Gambar 2.1 menunjukkan bentuk fisik, inti *die*, dan *schematic* dari Intel 4004.



(a) (b)

Gambar 2.1 (a) Intel 4004, (b) inti 4004

Mikroprosessor bekerja dengan cara menjalankan dan mengatur perpindahan data pada jalur data (data bus) dan serta memproses data yang didapatkan tergantung dari instruksi yang ada. Pengaturan dan pengkontrolan ini dilakukan dengan memanfaatkan jalur alamat dan jalur kontrol. Jalur kontrol berfungsi untuk membedakan kondisi perpindahan data (membaca atau menulis) dan sekaligus membedakan peripheral yang akan di kontrol (memori, rom,atau I/O dan lain sebagainya). Sedangkan jalur alamat berfungsi untuk mendefinisikan secara spesifik letak data yang akan diproses, seperti alamat dalam memori, I/O dan lain sebagainya.

Komponen mikroprosesor atau yang lebih mudah kita sebut dengan CPU (*central processing unit)* terbagi menjadi beberapa macam, yaitu sebagai berikut:

2.1.1 Unit Kontrol

Unit kontrol yang mampu mengatur jalannya program. Komponen ini sudah pasti terdapat dalam semua CPU.CPU bertugas mengontrol komputer sehingga terjadi sinkronisasi kerja antar komponen dalam menjalankan fungsi-fungsi operasinya. termasuk dalam tanggung jawab unit kontrol adalah mengambil intruksi-intruksi dari memori utama dan menentukan jenis instruksi tersebut. Bila ada instruksi untuk perhitungan aritmatika atau perbandingan logika, maka unit kendali akan mengirim instruksi tersebut ke ALU. Hasil dari pengolahan data dibawa oleh unit kendali ke memori utama lagi untuk disimpan, dan pada saatnya akan disajikan ke alat *output*. Dengan demikian tugas dari unit kendali ini adalah:

1) Mengatur dan mengendalikan alat-alat input dan output.
2) Mengambil instruksi-instruksi dari memori utama.
3) Mengambil data dari memori utama (jika diperlukan) untuk diproses.
4) Mengirim instruksi ke ALU bila ada perhitungan aritmatika atau perbandingan logika serta mengawasi kerja dari ALU.
5) Menyimpan hasil proses ke memori utama.

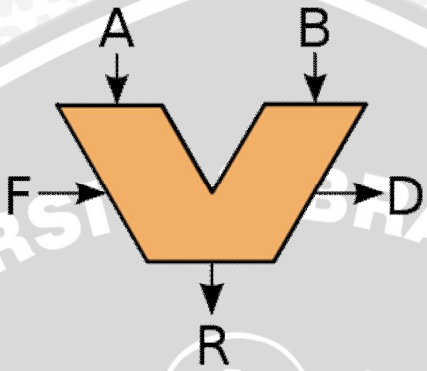
2.1.2 Register

Register merupakan alat penyimpanan kecil yang mempunyai kecepatan akses cukup tinggi, yang digunakan untuk menyimpan data dan/atau instruksi yang sedang diproses. Memori ini bersifat sementara, biasanya digunakan untuk menyimpan data saat di olah ataupun data untuk pengolahan selanjutnya. Secara analogi, register ini dapat diibaratkan sebagai ingatan di otak bila kita melakukan pengolahan data secara manual, sehingga otak dapat diibaratkan sebagai CPU, yang berisi ingatan-ingatan,

satuan kendali yang mengatur seluruh kegiatan tubuh dan mempunyai tempat untuk melakukan perhitungan dan perbandingan logika.

2.1.3 ALU

ALU unit yang bertugas untuk melakukan operasi aritmetika dan operasi logika berdasar instruksi yang ditentukan. Bentuk ALU biasanya digambarkan dalam bentuk V, Gambar 2.2 menunjukkan simbol sebuah ALU.



Gambar 2.2 Simbol ALU

Sumber: http://upload.wikimedia.org/wikipedia/commons/thumb/8/82/ALU_symbol.svg/300px-ALU_symbol.svg.png

ALU sering di sebut mesin bahasa karena bagian ini ALU terdiri dari dua bagian, yaitu unit aritmatika memiliki spesifikasi tugas tersendiri. Tugas utama dari ALU adalah melakukan semua perhitungan aritmatika (matematika) yang terjadi sesuai dengan instruksi program. ALU melakukan semua operasi aritmatika dengan dasar penjumlahan sehingga sirkuit elektronik yang digunakan disebut adder.

Cara Kerja CPU

Saat data dan/atau instruksi dimasukkan ke *processing-devices*, pertama sekali diletakkan di RAM (melalui *Input-storage*); apabila berbentuk instruksi ditampung oleh *Control Unit* di *Program-storage*, namun apabila berbentuk data ditampung di *Working-storage*). Jika register siap untuk menerima pengerjaan eksekusi, maka Control Unit akan mengambil instruksi dari Program-storage untuk ditampungkan ke Instruction Register, sedangkan alamat memori yang berisikan instruksi tersebut ditampung di *Program Counter*. Sedangkan data diambil oleh Control Unit dari *Working-storage* untuk ditampung di *General-purpose register* (dalam hal ini di *Operand-register*). Jika berdasar instruksi pengerjaan yang dilakukan adalah arithmatika dan logika, maka ALU akan mengambil alih operasi untuk mengerjakan berdasar instruksi yang ditetapkan. Hasilnya ditampung di *Accumulator*. Apabila hasil pengolahan telah selesai, maka

Control Unit akan mengambil hasil pengolahan di *Accumulator* untuk ditampung kembali ke *Working-storage*. Jika pengerjaan keseluruhan telah selesai, maka Control Unit akan menjemput hasil pengolahan dari *Working-storage* untuk ditampung ke Output-storage. Lalu selanjutnya dari *Outputstorage*, hasil pengolahan akan ditampilkan ke *output-devices*.

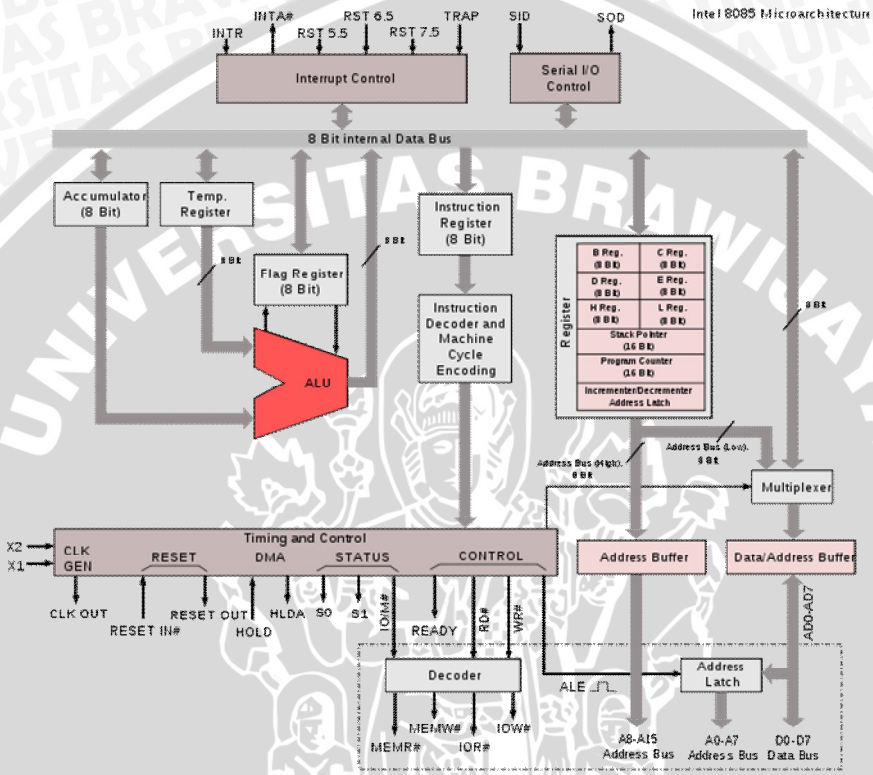
Fungsi CPU

CPU berfungsi seperti kalkulator, hanya saja CPU jauh lebih kuat daya pemrosesannya. Fungsi utama dari CPU adalah melakukan operasi aritmatika dan logika terhadap data yang diambil dari memori atau dari informasi yang dimasukkan melalui beberapa perangkat keras, seperti papan ketik, pemindai, tuas kontrol, maupun tetikus. CPU dikontrol menggunakan sekumpulan instruksi perangkat lunak komputer. Perangkat lunak tersebut dapat dijalankan oleh CPU dengan membacanya dari media penyimpan, seperti cakram keras, disket, cakram padat, maupun pita perekam. Instruksi-instruksi tersebut kemudian disimpan terlebih dahulu pada memori fisik (RAM), yang mana setiap instruksi akan diberi alamat unik yang disebut alamat memori. Selanjutnya, CPU dapat mengakses data-data pada RAM dengan menentukan alamat data yang dikehendaki.

Saat sebuah program dieksekusi, data mengalir dari RAM ke sebuah unit yang disebut dengan bus, yang menghubungkan antara CPU dengan RAM. Data kemudian didekode dengan menggunakan unit proses yang disebut sebagai pendekoder instruksi yang sanggup menerjemahkan instruksi. Data kemudian berjalan ke unit aritmatika dan logika (ALU) yang melakukan kalkulasi dan perbandingan. Data bisa jadi disimpan sementara oleh ALU dalam sebuah lokasi memori yang disebut dengan register supaya dapat diambil kembali dengan cepat untuk diolah. ALU dapat melakukan operasi-operasi tertentu, meliputi penjumlahan, perkalian, pengurangan, pengujian kondisi terhadap data dalam *register*, hingga mengirimkan hasil pemrosesannya kembali ke memori fisik, media penyimpan, atau register apabila akan mengolah hasil pemrosesan lagi. Selama proses ini terjadi, sebuah unit dalam CPU yang disebut dengan penghitung program akan memantau instruksi yang sukses dijalankan supaya instruksi tersebut dapat dieksekusi dengan urutan yang benar dan sesuai.

2.2 Mikroprosesor 8085

Mikroprosesor 8085 merupakan mikroprosesor 8bit produksi INTEL, diluncurkan sekitar tahun 1970. Pada zamannya sempat dijadikan sebagai mikroprosesor standar untuk sistem operasi. Arsitektur mikroprosesor INTEL 8085 diperlihatkan dalam Gambar 2.3.



Gambar 2.3 Arsitektur Mikroprosesor 8085

Mikroprosesor 8085 ini dikemas dalam bentuk DIP (Dual Inline Package) dengan jumlah penyemat sebanyak 40 buah. Dibandingkan pendahulunya 8080, mikroprosesor 8085 hanya membutuhkan sumber tegangan.

Mikroprosesor 8085 memiliki jumlah bus alamat sebanyak 16bit dengan demikian dapat mengakses memori secara langsung sebanyak 216 alamat memori atau sebanyak 65535 alamat, sering disebut sebagai 64K x 8Bit atau 64KByte memori secara langsung. Jumlah bus data adalah 8 bit, dengan demikian dapat menghubungi peripheral dengan lebar data (Data Path Width) 8bit.

Mikroprosesor ini tidak memiliki bus alamat 16bit secara terpisah, tetapi bus alamat byte terendah (low significant byte) yaitu A0..A7 dimultiplek dengan Bus Data D0..D7. Dengan demikian mikroprosesor 8085 belum siap dijadikan sebagai Unit

Mikroprosesor (MPU - Microprocessor Unit). Selain itu bus kontrol peripheral /MEMR, /MEMW, /IOR, dan /IOW belum terpisah sepenuhnya dan harus dibangkitkan dari sinyal kontrol /RD (penyemat 32), /WR (penyemat 31), IO/M (penyemat 34). Agar siap dijadikan sebagai MPU diperlukan beberapa komponen tambahan yaitu rangkaian bus demultiplexer (pemisahan bus D0..D7 dan A0..A7) dan rangkaian dekoder sinyal kontrol.

Register yang dimiliki mikroprosesor 8085 adalah sebagai berikut.

1) Register Akumulator (register A) yang mempunyai kapasitas 8 bit.
2) Register B, C, D, E, H, L. Dapat diperlakukan sebagai register 8bit atau pasangan register 8 bit menjadi 16 bit yaitu BC, DE dan HL. Register ini disebut sebagai User Register artinya register yang diperuntukkan pengguna agar bisa dilibatkan dalam pemrograman.
3) Register PC (Program Counter), berfungsi sebagai pointer alamat program yang akan dieksekusi.

2.2.1 Siklus Prosesor

Waktu yang diperlukan 8085 untuk menangkap dan mengeksekusi satu instruksi bahasa mesin disebut siklus instruksi. Tiap-tiap instruksi memiliki kekompleksan yang berbeda-beda dengan konsekuensi semakin kompleks instruksi maka semakin panjang pula waktu untuk mengeksekusi. Metode 8085 untuk mengeksekusi instruksi di dalam mikroprosesor terorganisir sehingga waktu yang dibutuhkan untuk mengeksekusi masing-masing instruksi lebih dapat diprediksi dan lebih umum.

Masing-masing instruksi dibagi ke dalam satu hingga lima siklus mesin. Masing-masing siklus mesin pada dasarnya hasil dari yang diperlukan, oleh instruksi yang dieksekusi, untuk mengakses memori. Instruksi tersingkat dapat membutuhkan hanya satu siklus mesin., dimana instruksi tersebut diperoleh dari memori. Siklus terpanjang, yang membutuhkan lima buah siklus mesin, dapat mengakses memori sebanyak 5 kali akses, akses pertama untuk memperoleh 1 byte instruksi itu sendiri, siklus kedua dan ketiga diperlukan untuk menangkap dua byte dari suatu alamat, sedangkan siklus keempat dan kelima dibutuhkan untuk menyimpan 2 byte alamat dalam memori.

Tipe siklus mesin yang dieksekusi telah dispesifikasi berdasar status IO/M, S0, dan S1, serta jalur kontrol yaitu, /RD, /WR, dan /INTA. Keenam jalur ini dapat menetapkan tujuh tipe siklus mesin yang berbeda seperti yang dijelaskan sebagai berikut.

1) Pengambilan op-code (*Op-code Fetch*)

   Op-code fetch adalah siklus mesin yang pertama untuk setiap instruksi. Proses ini diinisialisasikan dengan S0 dan S1 berlogika tinggi, dengan IO/M dan /RD berlogika rendah. Proses yang terjadi adalah siklus pembacaan memori untuk memperoleh satu byte instruksi.

2) Pembacaan memori (*Memory Read*)

   Siklus ini adalah siklus pembacaan byte lain kecuali op-code. Proses ini diinisialisasikan dengan S0 diset 0 dan S1 diset 1, dan IO/M dan /RD berlogika rendah. Proses yang terjadi adalah siklus pembacaan memori untuk memperoleh satu byte data atau alamat.

3) Penulisan Memori (*Memory Write*)

   Siklus ini adalah siklus normal penulisan ke memori. Proses ini diinisialisasikan dengan S0 diset 1, S1 diset 0, dan IO/M dan /WR berlogika rendah. Proses yang terjadi adalah siklus penulisan ke memori untuk menyimpan satu byte data pada suatu alamat.

4) Pembacaan Input/Output (*I/O Read*)

   Siklus ini adalah siklus normal pembacaan suatu devais I/O. Proses ini diinisialisasikan dengan S0 diset 0, S1 diset 1, dan IO/M berlogika tinggi dan /RD berlogika rendah. Proses yang terjadi adalah siklus pembacaan ang akan membawa satu byte dari suatu devais input ke dalam mikroprosesor.

5) Penulisan Input/Output (*I/O Write*)

   Siklus ini adalah siklus normal penulisan ke I/O. Proses ini diinisialisasikan denagan S0 diset 1, S1 diset 0, dan IO/M dan /WR berlogika rendah. Proses yang terjadi adalah siklus penulisan yang akan mengirimkan satu byte keluar dari mikroprosesor ke suatu devais output.

2.3 Memori

Memori adalah pusat kegiatan pada sebuah komputer, karena setiap proses yang akan dijalankan harus melalui memori terlebih dahulu. CPU mengambil instruksi dari memori sesuai dengan yang ada pada *program counter*. Tugas sistem operasi adalah mengatur peletakan banyak proses pada suatu memori.

Sebelum masuk ke memori, suatu proses harus menunggu di sebuah *input queue*, setelah itu barulah mereka akan diberikan alamat pada memori. Pemberian alamat dapat dilakukan pada waktu *compile*, waktu pemanggilan, dan waktu eksekusi.

Alamat logika (virtual) adalah alamat yang dibentuk di CPU, sedangkan alamat fisik adalah alamat yang terlihat oleh memori. Seluruh proses dan data berada dalam memori ketika dieksekusi. Berhubung ukuran dari memori fisik terbatas, kita harus melakukan pemanggilan dinamis untuk mendapatkan utilisasi ruang memori yang baik.

2.4 Field-Programmable Gate Array (FPGA)

FPGA adalah sebuah *integrated circuit* yang didesain untuk dapat dikonfigurasi oleh user atau designer setelah keluar dari produksi. Pengkonfigurasian FPGA pada umumnya adalah spesifik menggunakan deskripsi bahasa hardware atau HDL (*hardware description language*). FPGA dapat digunakan untuk diimplementasikan pada semua jenis fungsi logika yang terdapat atau yang dapat dilakukan oleh ASIC (*application specific integrated circuit* atau IC yang dibuat atau diciptakan secara khusus hanya untuk fungsi yang spesifik). Kemampuan FPGA dalam pemprograman ulang setelah *shipping* dan kebutuhan biaya yang relative lebih kecil dibanding ASIC menyebabkan FPGA merupakan sebuah keuntungan pada bermacam-macam aplikasi. Gambar 2.4 menujukkan contoh fisik sebuah IC FPGA.

Gambar 2.4 Bentuk fisik FPGA
Sumber : Altera Stratix IV EP4SGX230 FPGA on a PCB

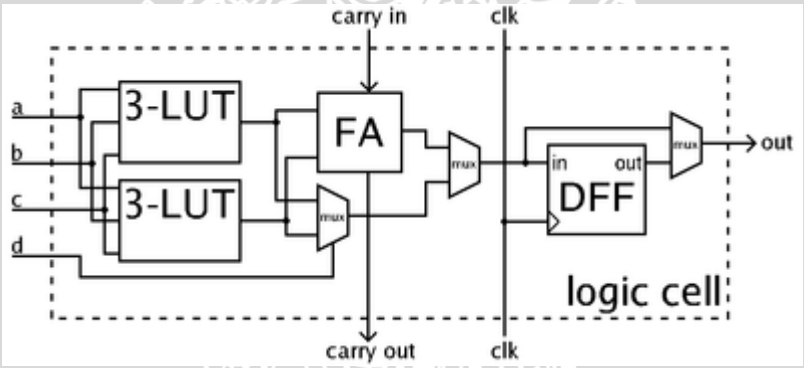
FPGA terdiri atas kumpulan komponen logika yang dapat diprogram ulang yang disebut dengan "*logic block*", dan sebuah hirarki *reconfigurable interconnect* (koneksi jalur-jalur atau penjaluran yang dapat diatur-atur hubungannya antara jalur satu dan yang lain) sehingga memungkinkan adanya hubungan blok-blok yang saling terkoneksi (*wired together*). Setiap *logic block*, dapat dikonfigurasi menjadi fungsi kombinasional yang kompleks, atau hanya menjadi sebuah gerbang sederhana (seperti gerbang AND

dan XOR). Pada sebagian besar FPGA, dalam setiap logic block terdapat juga elemen-elemen memori seperti flip-flop atau memori blok yang lebih kompleks.

Pada awalnya, FPGA merupakan pertunasan dari *programmable read-only memory* (PROM) dan *programmable logic device* (PLD), dimana kedua-duanya merupakan device yang memiliki kemampuan untuk diprogram ulang, walaupun pemprogramman yang ada adalah *hard-wired*.

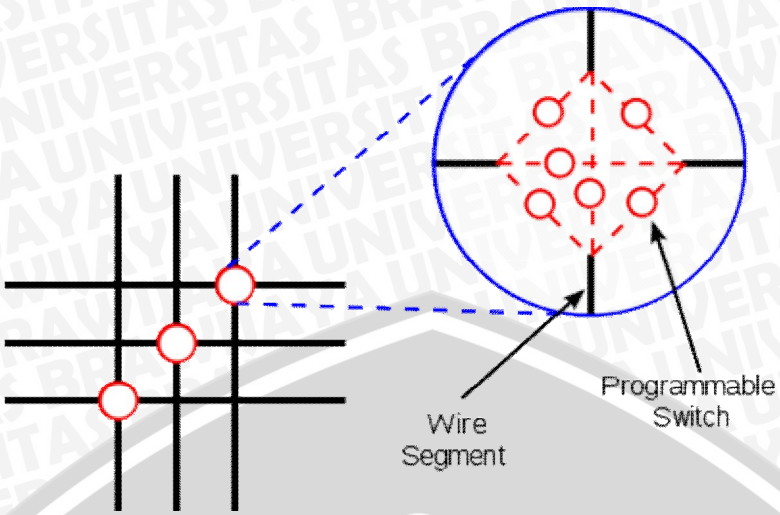
2.4.1 Arsitektur FPGA

Dalam sebagian besar arsitektur FPGA, terbentuk dari sebuah *logic block array* (disebut sebagai *configurable logic block* atau CLB), pad I/O, dan chanel-chanel penjaluran. Pada umumnya, dalam sebuah logic block terdapat beberapa sel logika. Sebuah sel khusus dapat terdiri dari 4-input *lookup table*, sebuah *full adder* dan d flip-flop. Masing-masing masukan dan keluaran setiap bagian dari sebuah sel, akan dihubungkan dengan *multiplekser*, sehingga sebuah sel dapat berdiri sendiri layaknya sebuah *lookup table, full adder* atau sebuah flip-flop, hingga menjadi kombinasi penuh antara *lookup table, full adder* dan flip-flop. Gambar 2.5 menunjukkan contoh *schematic* sebuah sel logika.



Gambar 2.5 Contoh Sebuah Sel Logika
Sumber: Xilinx documentation, ug070

Dalam chanel penjaluran (*routing channel*) sebuah FPGA, terdapat sebuah topologi pengaturan jalur yang dapat diprogram (*switch box topology*). Dalam kondisi awal, semua jalur tidak tersegmen yang artinya setiap jalur hanya terhubung dengan sebuah blok logika dan tidak terhubung dengan yang lain. Namun dengan mengaktifkan beberapa *programable switch* pada sistem topologi pengaturan FPGA, maka jalur-jalur dalam FPGA dapat terbentuk. Dengan demikian maka sebuah arsitektur IC yang baru dapat terbentuk, baik dengan harus mengatur jalur penghubung antar blok maupun pengaturan blok itu sendiri. Gambar 2.6 menunjukkan topologi *switch box*.



Gambar 2.6 Topologi Switch Box
Sumber: Xilinx documentation, ug070

Dalam FPGA yang telah modern, keluarga FPGA melakukan perluasan diatas kemampuan standarnya, yaitu dengan cara menambahkan fungsi-fungsi fix tingkat tinggi ke dalam bentuk *silicon*. Dengan demikian, FPGA ini dapat melakukan fungsi-fungsi yang lebih cepat jika dibandingkan dengan membuat fungsi yang telah fix tersebut dari blok-blok standart yang ada. Beberapa contoh dari fungsi fix tersebut antara lain *hardware multiplier*, blok *generic DSP*, sebuah *embedded processor*, I/O logic berkecepatan tinggi dan *embedded memory*.

BAB III
METODOLOGI

Penyusunan skripsi ini didasarkan pada masalah yang bersifat teoritis aplikatif, yaitu integrasi perancangan sistem mikroprosesor 8085 dan memori berdasarkan pada sistem-sistem yang telah ada dan merancangnya ke dalam bahasa hardware yang kemudian di aplikasikan ke dalam FPGA. Sistem arsitektur mikroprosesor 8085 yang telah ada merupakan data utama yang kemudian akan dianalisis dan dirancang kembali sesuai dengan spesifikasi yang ditentukan.

Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat dalam bab pendahuluan maka diperlukan metode untuk menyelesaikan masalah tersebut.

3.1 Studi Literatur

Literatur yang dibutuhkan adalah dasar teori yang berhubungan dengan sistem yang akan dirancang, yaitu sebagai berikut:

1) Arsitektur Mikroprosesor
2) Arsitektur Mikroprosesor 8085
3) Arsitektur Memori
4) Perangkat Modul Praktikum Mikroprosesor 8085
5) *Field Programmable Gate* Array (*FPGA*)

3.2 Penentuan Spesifikasi Rancangan

Spesifikasi rancangan secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi rancangan yang direncanakan adalah sebagai berikut:

1) Dapat mengakses perangkat tambahan berupa memori dan unit input output.
2) Terdapat sistem clock yang memberikan pulsa untuk menjalankan seluruh proses dalam sistem.
3) Pengaturan siklus instruksi, eksekusi, dan seluruh proses kerja mikroprosesor dikendalikan oleh unit kontrol yang memiliki sinyal kontrol ALE, IO/M', WR', dan RD'.
4) Memiliki program counter yang digunakan untuk mencacah alamat dan menset alamat utama saat terdapat instruksi jump.

5) Memiliki 7 buah register 8 bit yaitu register akumulator (A), B, C, D, E, H, dan L. Enam register yang disebutkan terakhir dapat digunakan berpasangan sehingga menjadi register pasangan 16 bit.
6) Dapat melakukan operasi aritmatika dan logika meliputi proses penjumlahan, pengurangan, OR, AND, dan XOR 2 buah data 8 bit yang diproses di dalam ALU.
7) Memiliki flag sebagai status hasil operasi yang telah dilakukan ALU meliputi Zero (Z), Carry (CY), Sign (S), Parity (P), dan Auxiliary Carry (AC).
8) Unit input dan output yang dirancang sebagai unit tambahan adalah unit yang memilik 8 jalur data dan 8 jalur alamat. Unit input ditetapkan beralamat 12H sedangkan unit output ditetapkan beralamat 13H.
9) Memori yang dirancang sebagai unit tambahan adalah memori yang memiliki 8 jalur data dan dapat diakses dengan 16 jalur data sesuai spesifikasi mikroprosesor 8085. Memori ini mempunyai rentang alamat 0000H-00FFH.
10) Sistem ini dirancang dalam FPGA XILINX tipe SPARTAN XC3S500E.

3.3 Perancangan dan Perealisasian Sistem

Perancangan dan perealisasian sistem dilakukan dengan urutan sebagai berikut:

1) Pembuatan diagram blok mikroprosesor, unit untuk menjembatani dengan unit luar beserta jalur-jalurnya.
2) Perealisasian diagram blok sistem menggunakan bahasa VHDL menggunakan software Xilinx ISE Project Navigator 11.
3) Penerapan dalam FPGA.

Switch sebagai input dan LED sebagai output yang telah ada dalam FPGA yang digunakan tidak mencukupi untuk sistem yang dirancang sehingga membutuhkan unit input dan output tambahan yang didesain menggunakan software Eagle Layout Editor yang hasilnya dicetak pada *Printed Circuit Board* (*PCB*).

3.4 Pengujian

Pengujian dilakukan bersamaan dengan penulisan bahasa *hardware* ke dalam modul FPGA, Hal ini dilakukan guna memberikan hasil rancangan mendekati dengan sistem yang telah direncanakan dan menanggulangi keterbatasan modul FPGA dalam penerapan rancangan yang telah dibuat. Pengujian dilakukan dengan tahap pengujian tiap blok kemudian pengujian keseluruhan sistem.

3.4.1 Pengujian Tiap Blok

Pengujian tiap blok yang dilakukan adalah sebagai berikut.

1) Sistem Clock

   Pengujian sistem timing dan clock dilakukan untuk mengetahui apakah clock yang digunakan telah sesuai dengan spesifikasi mikroprosesor 8085.

2) Input dan Output

   Pengujian terhadap input dan output ini dilakukan untuk mengetahui apakah sistem masukan dapat menerima dan menyimpan perintah yang diberikan oleh user dan untuk mengetahui apakah sistem keluaran display dapat menampilkan hasil yang sesuai.

3) Memori

   Pengujian terhadap memori dilakukan untuk mengetahui apakah memori dapat ditulis atau dibaca dengan baik sesuai dengan alamat yang diberikan oleh mikroprosesor dan sinyal kontrol yang diterima oleh memori dengan jalur alamat 16 bit dan jalur data 8 bit.

4) Blok Bidirectional, Buffer, dan Dekoder

   Pengujian terhadap blok ini digunakan untuk mengetahui apakah blok untuk menjembatani antara mikroprosesor dengan unit input output dan memori dapat menerjemahkan sinyal-sinyal dan mentransfer data dengan tepat.

5) ALU

   Pengujian terhadap ALU dilakukan untuk mengetahui apakah ALU dapat memproses penjumlahan, pengurangan, OR, AND, XOR, dan logika lain sesuai dengan perintah yang telah diterjemahkan dalam dekonder instruksi kemudian diperintahkan oleh unit kontrol.

4) Flag

   Flag diuji bersamaan dengan pengujian ALU untuk diketahui apakah keluaran status flag sesuai dengan hasil operasi yang telah diproses oleh ALU.

3.4.2 Pengujian Keseluruhan Sistem

Pengujian keseluruhan sistem dilakukan setelah pengujian tiap blok selesai. Pengujian ini dibutuhkan untuk mengetahui apakah sistem yang telah dirancang sesuai dengan spesifikasi rancangan yang telah ditentukan. Pengujian ini dilakukan dengan menggabungkan blok-blok yang telah dirancang yang dituliskan kembali ke dalam

FPGA. Pengujian dilakukan dengan memberikan masukan ke dalam sistem dan melihat apakah hasil yang ditunjukkan sesuai dengan yang diinginkan baik berupa hasil keluaran display output LED maupun berupa status keluaran logika berupa diagram waktu.

BAB IV
PERANCANGAN

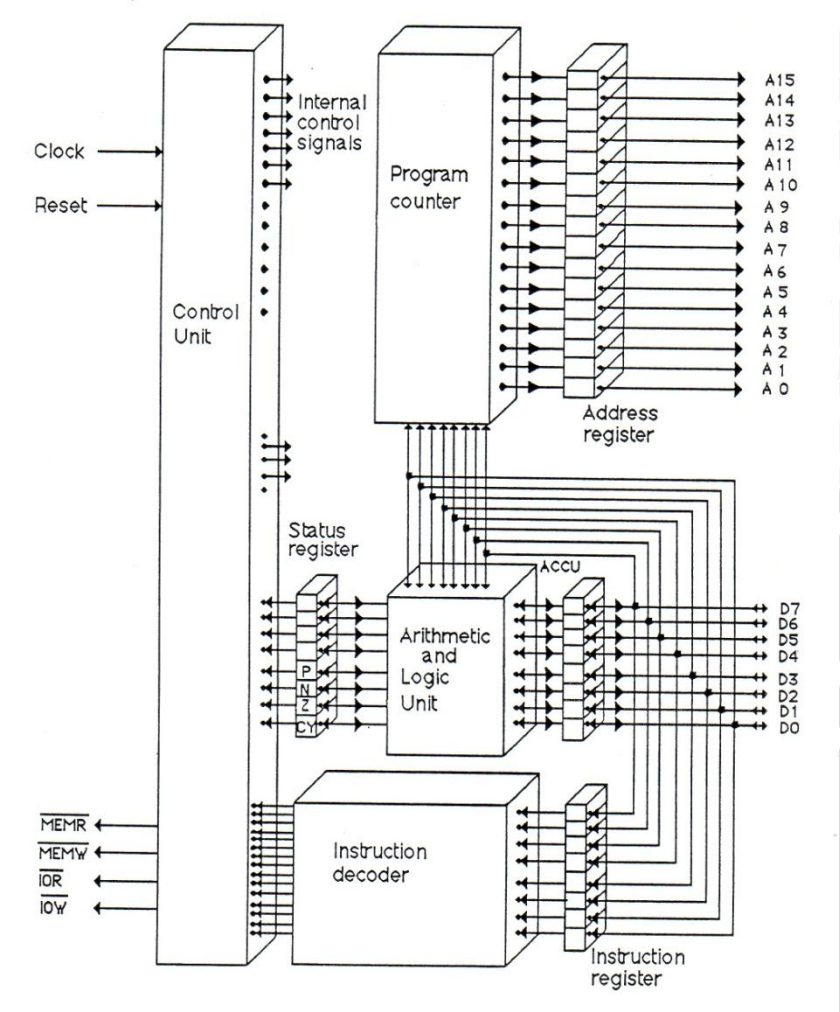
Bab ini membahas tentang perancangan sistem mikroprosesor 8085 beserta sistem memori yang kompatibel dengan mikroprosesor 8085. Sistem yang akan diimplementasikan dalam FPGA (*Field Programmable Gate Array*) ini digunakan untuk menjalankan beberapa instruksi yang sesuai dengan set instruksi mikroprosesor 8085 dan dapat menampilkan diagram waktu proses jalannya siklus mesin mikroprosesor tersebut. Perancangan ini meliputi perancangan sistem mikroprosesor, jalannya sistem mikroprosesor beserta koordinasi mikroprosesor dengan unit-unit tambahan, serta perancangan tiap blok unit-unit penyusun sistem mikroprosesor, sistem bidirectional buffer dekoder, memori, serta input output. Dalam perancangan ini dilakukan dalam 3 tahap. Tahap pertama adalah tahap perancangan gambaran cara kerja sistem digital. Tahap kedua adalah penetapan set instruksi yang dapat dijalankan sistem. Tahap ketiga adalah tahap perancangan tiap blok digital yang dibutuhkan.

4.1 Gambaran Kerja Sistem

Berdasarkan prinsip kerja mikroprosesor 8085 dan sistem memori yang telah dijelaskan dalam bab kajian pustaka,batasan masalah yang telah disebutkan dalam bab pendahuluan, serta spesifikasi perancangan sistem pada bab metodologi, perancangan gambaran cara kerja sistem dilakukan agar sesuai dengan karakteristik yang diinginkan.

4.1.1 Arsitektur Internal CPU

Arsitektur internal CPU mikroprosesor 8085 terdiri atas unit kontrol yang berfungsi sebagai pengendali utama jalannya mikroprosesor yang membutuhkan *clock* sebagai pulsa timingnya, *program counter* yang menunjukkan alamat lokasi memori dimana data atau instruksi berikutnya disimpan, *address register* yang menunjukkan alamat memori yang akan diakses atau alamat port I/O yang akan dieksekusi. Selama siklus pengambilan, instruksi yang diambil disampaikan ke register intruksi yang kemudian akan dipecahkan di dalam dekoder instruksi. Informasi yang didapat dari hasil pemecahan dekoder instruksi dikirimkan ke dalam unit kontrol agar instruksi dapat dimengerti. Arsitektur Internal CPU ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Arsitektur Internal CPU

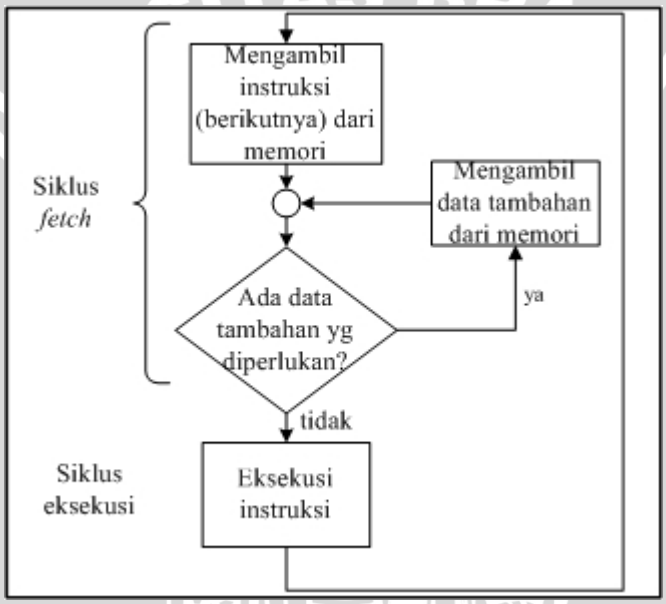
Jika terdapat instruksi yang mengandung operasi aritmatika dan logika maka unit ALU akan bekerja. ALU adalah unit yang digunakan untuk mengeksekusi operasi aritmatika dan logika. Jika hasil operasi ALU memiliki keadaan-keadaan istimewa, maka kondisi-kondisi tersebut dicantumkan dalam status register (register flag). Hasil dari operasi-operasi tersebut dapat disimpan dalam register ataupun dapat dikeluarkan ke unit output tergantung dari instruksi yang diberikan. Hubungan internal CPU dengan register dan unit luar akan dibahas dalam subbab selanjutnya.

4.1.2 Alur Pemrosesan Instruksi

Fungsi mikroprosesor terbagi dalam pengambilan dan pengeksekusian siklus masing-masing instruksi dalam program. Program yang dimaksud adalah sekumpulan instruksi yang disimpan dalam memori secara berurutan. Dalam proses normal operasi, mikroprosesor mengambil dan mengeksekusi satu instruksi dalam satu kurun waktu tertentu secara berurutan.

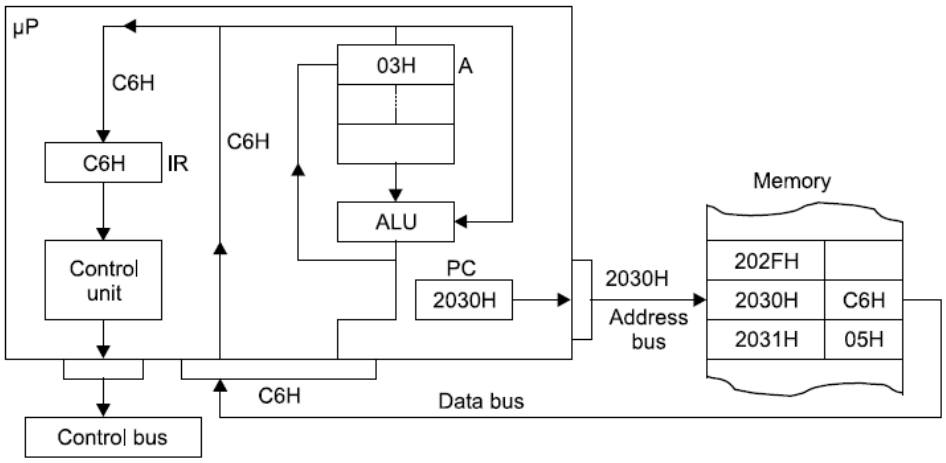
Tanpa memperhatikan tipe instruksi, mikroprosesor akan selalu memproses instruksi dengan jalan sebagai berikut:

a) Menggunakan sinyal kontrol MEMR untuk membaca instruksi dan menyimpannya dalam register instruksi,
b) Mikroprosesor akan mengambil operand dari memori atau menjalankan instruksi,tergantung dari instruksi yang dibaca,
c) Sekali instruksi dijalankan, mikroprosesor secara kontinyu mengalamati lokasi memori selanjutnya dengan program counter, kemudian membaca instruksi berikutnya. Alur pemprosesan instruksi ditunjukkan dalam Gambar 4.2.



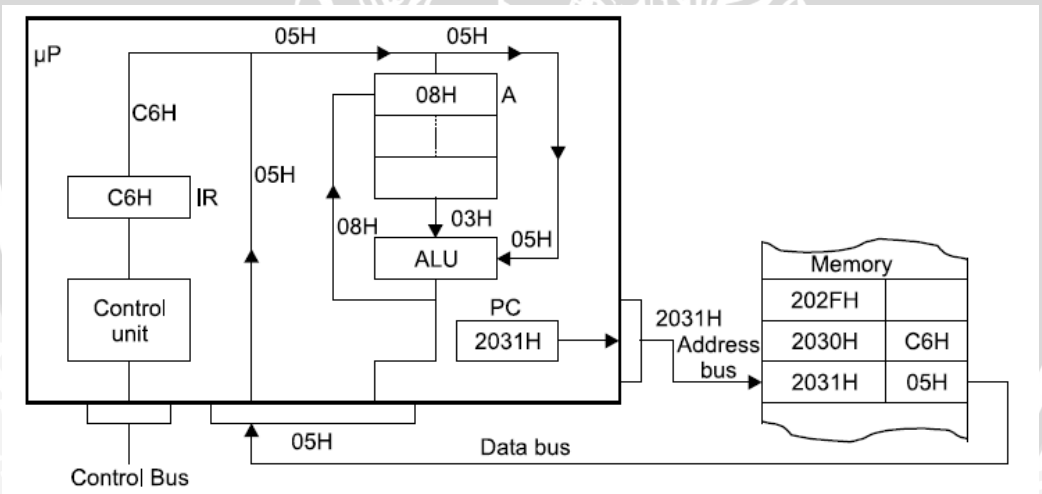
Gambar 4.2 Alur Pemrosesan Instruksi pada Mikroprosesor

Berikut ini akan dijelaskan contoh proses siklus fetch dalam mikroprosesor. Dalam penjelasan ini, yang akan dicontohkan adalah operasi aritmatika yang memerlukan data tambahan namun pengeksekusian terjadi dalam mikroprosesor. Instruksi yang akan dicontohkan adalah instruksi ADI 05H. Instruksi ini berarti menjumlahkan isi akumulator dengan bilangan 05H. Siklus dalam instruksi ini meliputi 2 tahap yaitu tahap pembacaan byte pertama yaitu pembacaan instruksi dan yang kedua adalah pembacaan operand sekaligus pengeksekusian yaitu penjumlahan bilangan 05H ke dalam akulumator. Siklus fetch ditunjukkan dalam Gambar 4.3.



Gambar 4.3 Siklus Fetch: Peletakan Byte Pertama (Opcode) kedalam Register Instruksi

Bagian pengambilan atau siklus fetch instruksi sama untuk tiap-tiap instruksi. Unit kontrol meletakkan isi program counter 2030H pada bus alamat. Byte pertama (dalam contoh ini adalah opcode 06H) diletakkan kedalam register instruksi (telah dijelaskan diatas bahwa proses ini menggunakan sinyal MEMR). Untuk proses pengeksekusian dijelaskan dalam Gambar 4.4.



Gambar 4.4 Pengeksekusian Instruksi: Pembacaan Byte Kedua dari Memori dan Penambahan ke Akumulator

Dalam siklus eksekusi instruksi, unit kontrol memeriksa opcode sebagai penerjemahan operasi pembacaan atau penulisan pada memori yang akan dilakukan kemudian berdasar apakah data tambahan dibutuhkan atau tidak. Dalam kasus ini, data 05H dari memori ditrasnfer melalui bus data kedalam ALU. Dalam waktu yang sama unit kontrol mengirim isi akumulator (03H) ke dalam ALU dan melakukan operasi

penjumlahan. Hasil operasi penjumlahan adalah 08H diletakkan dalam akumulator dengan menumpuki data akumulator sebelumnya yaitu 03H. Dengan adanya penyelesaian satu instuksi maka program counter secara otomatis menambahkan satu untuk menunjukkan lokasi memori berikutnya untuk mengeksekusi instruksi berikutnya.

4.1.3 Hubungan Internal CPU, Register, dan Address Buffer dalam Mikroprosesor 8085

Pada bahasan sebelumnya telah dijelaskan tentang arsitektur internal CPU mikroprosesor 8085. Pada pembahasan ini yang akan dijelaskan adalah sistem internal CPU yang terhubung dengan register-register yang terdapat dalam mikroprosesor 8085. Register adalah unit penyimpan data sementara. Selain register akumulator dan *temporary register*, terdapat 6 register lain yang terdapat dalam mikroprosesor yaitu register B, register C, register D, register E, register H, dan register L. Register ini adalah register 8 bit namun register-register tersebut dapat digunakan secara berpasangan sehingga disebut *pair register* (Rp). Register-register tersebut adalah register BC, DE, dan HL yang berkapasitas 16 bit.

Dalam sistem yang dirancang ini, unit mikroprosesor siap untuk digunakan berhubungan dengan unit luar. Siklus yang akan dilakukan dilakukukan oleh sistem ini selalu diawali dari pengambilan op-code untuk instruksi yang telah tersimpan dalam memori. Pada perancangan ini tidak terdapat instruksi untuk pembacaan memori secara langsung selain pengambilan op-code sehingga kontrol yang diperlukan oleh sistem dan yang terhubung dengan unit luar adalah kontrol ALE, IO/M', WR', dan RD'.

ALE adalah kontrol untuk mengaktifkan *addres latch*. Saat ALE aktif, maka mikroprosesor akan mengirimkan alamat untuk menunjuk unit luar yang dibutuhkan baik alamat memori maupun unit input ataupun output.

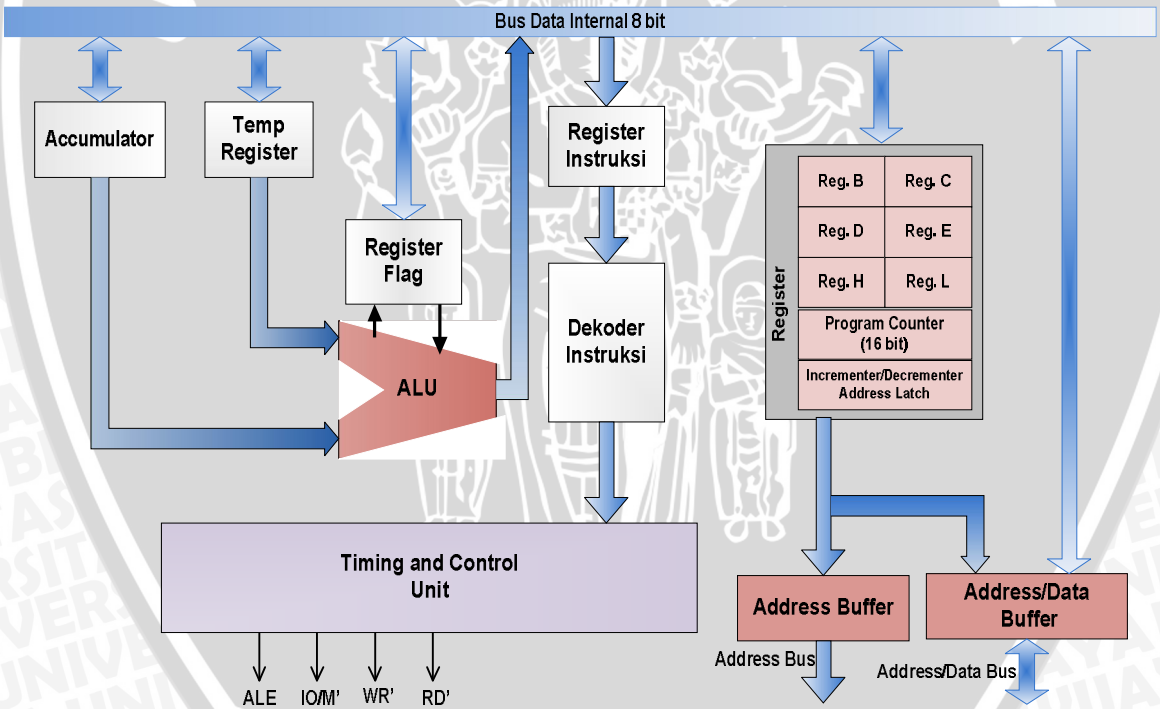
IO/M' adalah kontrol yang memilih kapan saatnya mikroprosesor mengakses unit input ataupun output dan kapan saatnya mikroprosesor mengakses memori. Mikroprosesor akan mengakses unit input output saat berlogika 1 sedangkan saat berlogika 0 mikroprosesor akan mengakses memori.

WR' adalah kontrol yang mengirimkan sinyal agar mikroprosesor menuliskan data ke unit luar mikroprosesor sedangkan RD' adalah kontrol yang mengirimkan sinyal agar mikroprosesor membaca data dari unit luar mikroprosesor.

Dibutuhkannya kontrol yang menghasilkan sinyal-sinyal untuk mikroprosesor mengakses dari atau keluar mikroprosesor itu dikarenakan mikroprosesor tidak dapat

bekerja sendiri. Mikroprosesor tetap membutuhkan unit luar sebagai penunjang jalannya sistem. Oleh karena itu, sistem mikroprosesor yang dirancang memiliki jalur yang digunakan untuk menghubungkan mikroprosesor 8085. Jalur tersebut adalah jalur alamat dan jalur data/alamat.

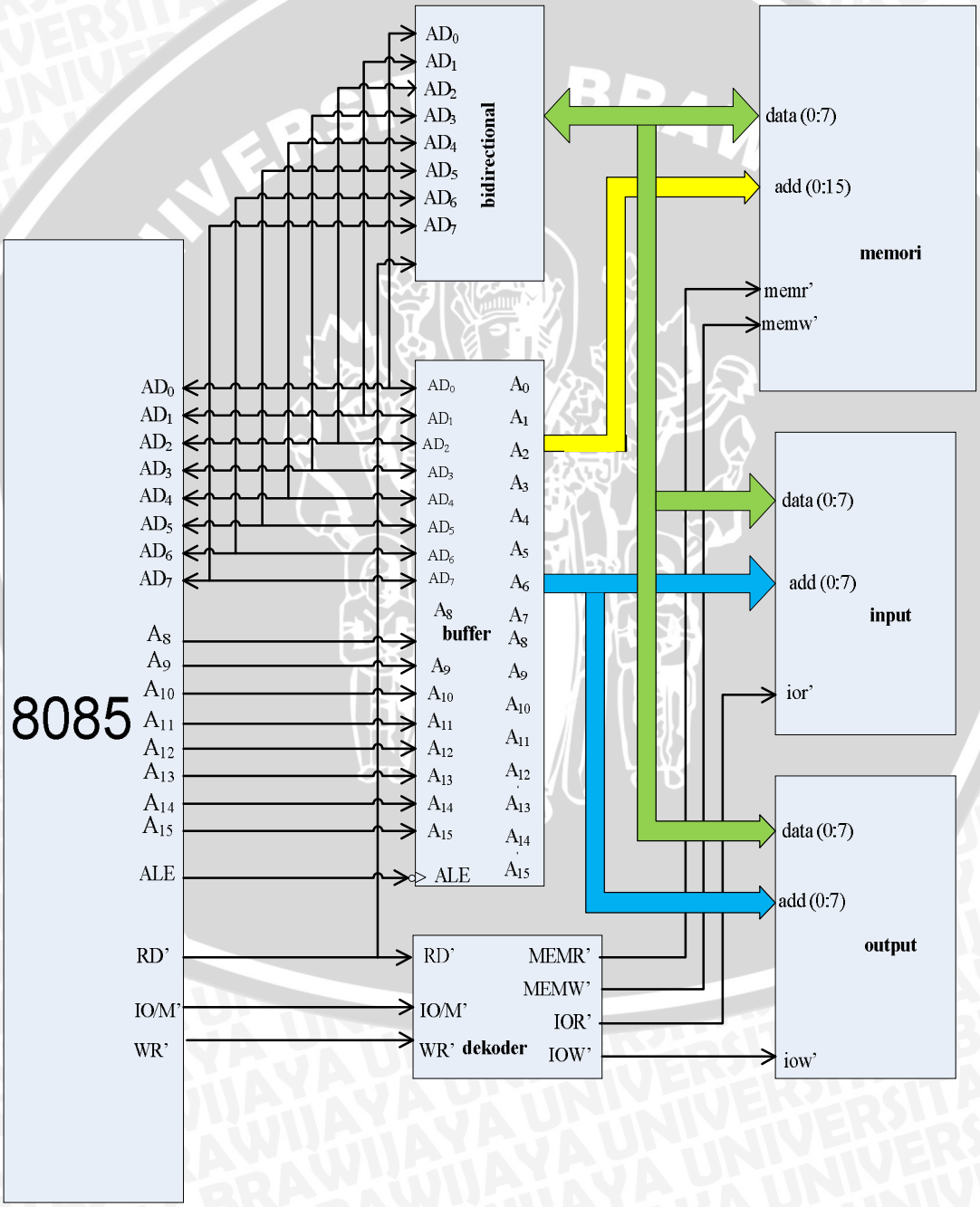
Jalur alamat pada mikroprosesor adalah jalur alamat untuk bit ke-8 hingga bit ke-15. Jalur alamat/data digunakan secara bersama-sama bergantung kontrol yang diberikan. Saat ALE aktif, maka jalur tersebut akan digunakan sebagai jalur alamat, sedangkan jika ALE tidak aktif, maka jalur tersebut digunakan sebagai jalur data. Jalur data bersifat bidirectional yang berarti dapat menjadi jalur data dua arah. Hal ini bergantung dari sinyal kontrol RD' dan WR'. Jika RD' aktif maka data akan mengarah dari luar ke dalam mikroprosesor sedangkan jika WR' aktif maka data akan mengarah ke luar mikroprosesor. Hubungan unit-unit dalam internal mikroprosesor 8085 ditunjukkan dalam Gambar 4.5.



Gambar 4.5 Arsitektur Internal Mikroprosesor 8085

4.1.4 Arsitektur Keseluruhan Sistem

Alur pemrosesan instruksi yang telah dijelaskan dalam sub-bab sebelumnya akan dapat dijalankan saat mikroprosesor 8085 terhubung dengan komponen tambahan salah satunya memori karena memori adalah tempat untuk menyimpan data yang akan diterjemahkan oleh mikroprosesor sebagai instruksi ataupun operand. Komponen lainnya yang dapat diintegrasikan dengan mikroprosesor adalah unit input dan output. Arsitektur keseluruhan sistem mikroprosesor 8085 yang dirancang ditunjukkan dalam Gambar 4.6.



Gambar 4.6 Keseluruhan Sistem Modul Mikroprosesor 8085

Dalam Gambar 4.6 ditunjukkan bahwa sistem keseluruhan ini meliputi mikroprosesor 8085, buffer, latch, dekoder, unit memori, unit input, dan unit output. Mikroprosesor 8085 dalam perancangan ini memiliki pin AD (0:7), A (0:7), IO/M', WR', dan RD', dan clock.

Pin AD(0:7) adalah pin address data merupakan pin yang dapat berfungsi bidirectional. Saat digunakan untuk mengirimkan alamat, pin ini berfungsi searah yaitu mengirimkan alamat (0:7) ke arah memori sedangkan saat digunakan untuk transfer data, maka pin ini dapat berfungsi untuk dua arah. Pin A(0:7) adalah pin address merupakan pin yang berfungsi untuk mengirimkan alamat (8:15) kea rah memori. Pin ini bersifat satu arah saja.

Pin IO/M' adalah pin yang mengontrol apakah mikroprosesor mengakses unit input output atau mengakses unit memori. Saat berlogika 1,mikroprosesor mengakses unit input output sedangkan saat berlogika 0, mikroprosesor mengakses unit memori. Pin WR' dan RD' adalah pin yang mengontrol apakah mikroprosesor mengakses komponen luar mikroprosesor dengan membaca atau menuliskan data. Saat WR' berlogika 0, mikroprosesor aktif menuliskan data ke komponen luar mikroprosesor. Penulisan ke unit memori atau ke unit input output bergantung pada logika pin IO/M. Sedangkan saat RD' berlogika 0 maka mikroprosesor aktif membaca data dari komponen luar mikroprosesor dan pembacaan dari unit memori atau dari unit output bergantung pada logika IO/M'. Clock berfungsi untuk mengatur keseluruhan clock yang dibutuhkan yang terdapat dalam sistem.

Unit buffer dalam sistem berfungsi untuk mengendalikan kapan saatnya data masuk atau keluar. Kontrol unit buffer didapatkan dari keluaran pin RD'. Unit latch berfungsi untuk mengendalikan kapan saatnya alamat terkirim pada unit yang dikehendaki untuk diakses. Unit latch dikendalikan oleh pin ALE.

Masukan unit dekoder terdiri atas IO/M, RD', dan WR' yang menghasilkan empat keluaran yaitu MEMR',MEMW', IOR', dan IOW'. Hasil keluaran dekoder inilah yang akan menentukan akses baca atau tulis dari atau ke unit input output maupun ke unit memori.

4.2 Instruksi-instruksi yang Dapat Dijalankan Sistem

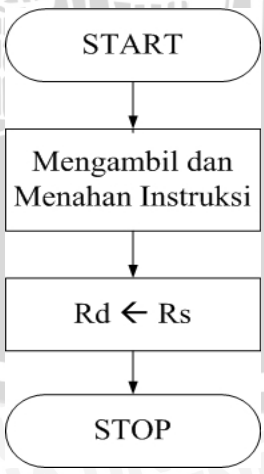
Dalam rancangan sistem mikroprosesor 8085 dan sistem memori ini, terdapat beberapa instruksi yang dapat dijalankan oleh sistem tersebut. Instruksi yang dapat dijalankan diambil dari set instruksi 8085 sehingga op-code dari tiap-tiap instruksi mengacu pada sumber tersebut. Instruksi-instruksi ini dipilih berdasarkan instruksi yang sering digunakan dalam praktikum mikroprosesor 8085. Instruksi-instruksi yang dapat dijalankan oleh sistem akan dijelaskan sebagai berikut.

4.2.1 Instruksi Transfer Data

Instruksi transfer data sebenarnya merupakan proses duplikasi data karena operasi tersebut bertujuan untuk menduplikasi data pada register sumber dan diletakkan di register tujuan. Proses ini lebih tepat dinamakan duplikasi karena proses pentransferan data ini tidak mengubah isi register sumber. Proses transfer data dapat terjadi diantara register dengan register, memori dengan register, register dengan data immediate, register dengan memori, memori dengan memori, dan memori dengan data immediate.

4.2.1.1 Instruksi MOV Rd,Rs

Instruksi MOV Rd,Rs adalah instruksi yang memerintahkan mikroprosesor untuk menduplikasi isi suatu register sumber ke dalam register target.Rd adalah register destinasi atau register target sedangkan Rs adalah register source atau register sumber. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi MOV ditunjukkan dalam Gambar 4.7.



Gambar 4.7 Diagram Alir Instruksi MOV

Jika dijabarkan dalam step-step, instruksi MOV dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Isi register sumber (Rs) diduplikasi kedalam register target (Rd)
4) Instruksi berakhir

Instruksi MOV Rd,Rs dan op-code masing-masing instruksi dijabarkan dalam Tabel 1.

Tabel 1 Instruksi MOV Rd,Rs dan Op-code

Instruksi	Opcode	Instruksi	Opcode
MOV A,A	7F	MOV C,A	4F
MOV A,B	78	MOV C,B	48
MOV A,C	79	MOV C,C	49
MOV A,D	7A	MOV C,D	4A
MOV A,E	7B	MOV C,E	4B
MOV A,H	7C	MOV C,H	4C
MOV A,L	7D	MOV C,L	4D

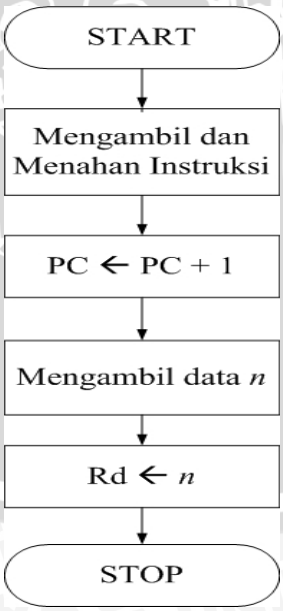
Instruksi	Opcode	Instruksi	Opcode
MOV B,A	47	MOV D,A	57
MOV B,B	40	MOV D,B	50
MOV B,C	41	MOV D,C	51
MOV B,D	42	MOV D,D	52
MOV B,E	43	MOV D,E	53
MOV B,H	44	MOV D,H	54
MOV B,L	45	MOV D,L	55

Instruksi	Opcode		
MOV E,A	5F	MOV E,D	5A
MOV E,B	58	MOV E,E	5B
MOV E,C	59	MOV E,H	5C
		MOV E,L	5D

Instruksi	Opcode	Instruksi	Opcode
MOV H,A	67	MOV L,A	6F
MOV H,B	60	MOV L,B	68
MOV H,C	61	MOV L,C	69
MOV H,D	62	MOV L,D	6A
MOV H,E	63	MOV L,E	6B
MOV H,H	64	MOV L,H	6C
MOV H,L	65	MOV L,L	6D

4.2.1.2 Instruksi MVI

Instruksi MVI adalah instruksi yang memerintahkan mikroprosesor untuk memasukkan suatu angka langsung ke register yang terdapat di dalam mikroprosesor. Dalam sistem yang dirancang ini, MVI hanya digunakan untuk memindahkan suatu nilai ke dalam register, tidak ke dalam memori sehingga penulisan instruksi ini adalah MVI Rd,n. Rd menunjukkan register target sedangkan n adalah nilai suatu angka yang akan dimasukkan. Instruksi ini adalah instruksi 2 byte yang memerlukan 2 kali siklus mesin dengan 7 T states. Diagram alir untuk instruksi MVI ditunjukkan dalam Gambar 4.8.



Gambar 4.8 Diagram Alir Instruksi MVI

Jika dijabarkan dalam step-step, instruksi MVI dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Mengambil angka dari memori
4) Memindahkan angka ke register yang dituju
5) Instruksi berakhir

Instruksi MVI Rd,n dan op-code masing-masing instruksi dijabarkan dalam Tabel 2.

Tabel 2 Instruksi MVI Rd,n dan Op-code

Instruksi	Opcode
MVI A,n	3E
MVI B,n	06
MVI C,n	0E
MVI D,n	16
MVI E,n	1E
MVI H,n	26
MVI L,n	2E

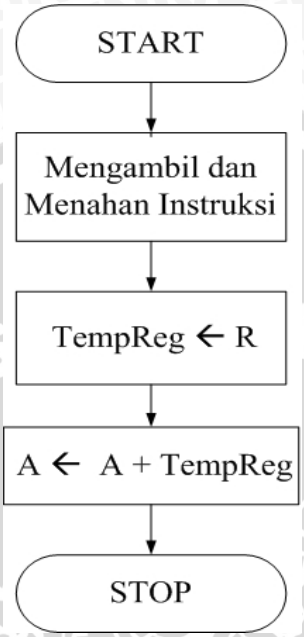
4.2.2 Instruksi Aritmatika dan Logika

Instruksi aritmatika meliputi instruksi penambahan dan pengurangan. Operasi penambahan ataupun pengurangan terdapat dua jenis yaitu operasi pengurangan atau penambahan dengan nilai satu dan operasi pengurangan atau penambahan dengan nilai lebih dari satu. Instruksi logika meliputi instruksi-instruksi logika dasar (OR,XOR,AND), logika pembanding, logika rotasi, dan instruksi komplemen. Seluruh intruksi aritmatika dan logika dioperasikan menggunakan unit ALU yaitu unit yang bertugas untuk mengoperasikan seluruh instruksi yang termasuk dalam instruksi aritmatika dan logika.

4.2.2.1 Instruksi ADD

Instruksi ADD adalah instruksi yang memerintahkan mikroprosesor untuk menambahkan register akumulator dengan angka yang terdapat dalam suatu register dan hasilnya dimasukkan ke dalam register akumulator. Nilai dalam status flag akan berubah untuk merefleksikan hasil penjumlahan dalam instruksi ini.

Dalam sistem yang dirancang ini, ADD digunakan untuk menjumlahkan suatu nilai dalam register saja sehingga penulisan instruksi ini adalah ADD R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi ADD ditunjukkan dalam Gambar 4.9.



Gambar 4.9 Diagram Alir Instruksi ADD

Jika dijabarkan dalam step-step, instruksi ADD dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register sementara dijumlahkan dengan isi register akumulator,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi ADD R dan op-code masing-masing instruksi dijabarkan dalam Tabel 3.

Tabel 3 Instruksi ADD dan Op-code

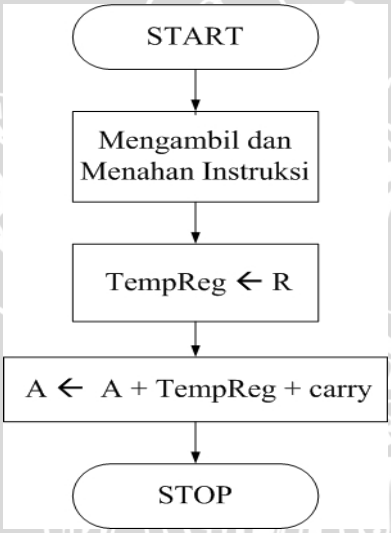
Instruksi	Opcode
ADD A	87
ADD B	80
ADD C	81
ADD D	82

ADD E	83
ADD H	84
ADD L	85

4.2.2.2 Instruksi ADC

Instruksi ADC adalah instruksi yang memerintahkan mikroprosesor untuk menjumlahkan isi register akumulator,carry,dan angka yang terdapat dalam suatu register dan hasilnya dimasukkan ke dalam register akumulator. Nilai dalam status flag akan berubah untuk merefleksikan hasil penjumlahan dalam instruksi ini.

Dalam sistem yang dirancang ini, ADC digunakan untuk menjumlahkan suatu nilai dalam register saja sehingga penulisan instruksi ini adalah ADC R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi ADC ditunjukkan dalam Gambar 4.10.



Gambar 4.10 Diagram Alir Instruksi ADC

Jika dijabarkan dalam step-step, instruksi ADC dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register sementara dijumlahkan dengan isi register akumulator dan carry,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi ADC R dan op-code masing-masing instruksi dijabarkan dalam Tabel 4.

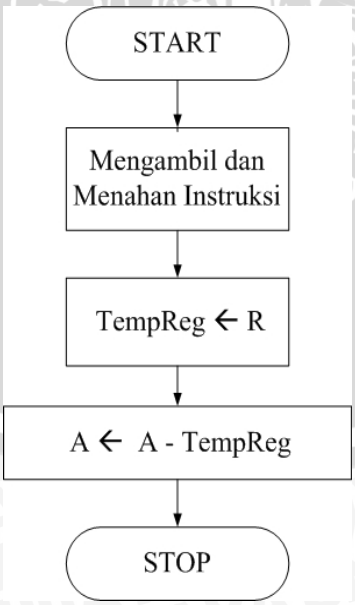
Tabel 4 Instruksi ADC dan Op-code

Instruksi	Opcode
ADC A	8F
ADC B	88
ADC C	89
ADC D	8A
ADC E	8B
ADC H	8C
ADC L	8D

4.2.2.3 Instruksi SUB

Instruksi SUB adalah instruksi yang memerintahkan mikroprosesor untuk mengurangkan isi register akumulator dengan angka yang terdapat dalam suatu register dan hasilnya dimasukkan ke dalam register akumulator. Nilai dalam status flag akan berubah untuk merefleksikan hasil penjumlahan dalam instruksi ini.

Dalam sistem yang dirancang ini, SUB digunakan untuk mengurangkan akumulator dengan suatu nilai dalam register saja sehingga penulisan instruksi ini adalah SUB R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi SUB ditunjukkan dalam Gambar 4.11.



Gambar 4.11 Diagram Alir Instruksi SUB

Jika dijabarkan dalam step-step, instruksi SUB dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register akumulator dikurangi dengan isi register sementara,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi SUB R dan op-code masing-masing instruksi dijabarkan dalam Tabel 5.

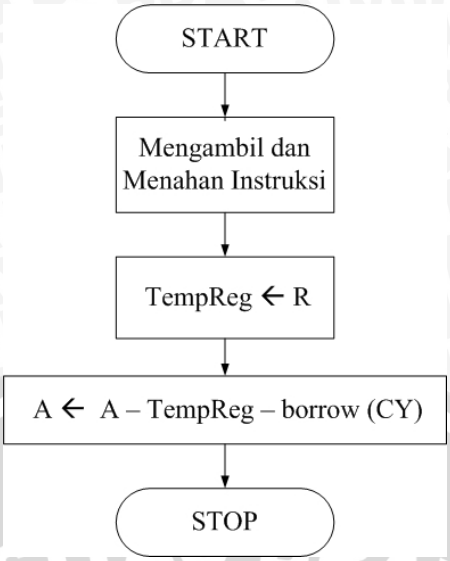
Tabel 5 Instruksi SUB dan Op-code

Instruksi	Opcode
SUB A	97
SUB B	90
SUB C	91
SUB D	92
SUB E	93
SUB H	94
SUB L	95

4.2.2.4 Instruksi SBB

Instruksi SBB adalah instruksi yang memerintahkan mikroprosesor untuk mengurangkan isi register akumulator dengan angka yang terdapat dalam suatu register dan borrow,kemudian hasilnya dimasukkan ke dalam register akumulator. Borrow diambil dari nilai yang terdapat dalam status flag carry (bit ke-0). Nilai dalam status flag akan berubah untuk merefleksikan hasil penjumlahan dalam instruksi ini.

Dalam sistem yang dirancang ini, SBB digunakan untuk mengurangkan akumulator dengan suatu nilai dalam register saja sehingga penulisan instruksi ini adalah SBB R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi SBB ditunjukkan dalam Gambar 4.12.



Gambar 4.12 Diagram Alir Instruksi SBB

Jika dijabarkan dalam step-step, instruksi SBB dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register akumulator dikurangi dengan isi register sementara dan borrow,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi SBB R dan op-code masing-masing instruksi dijabarkan dalam Tabel 6.

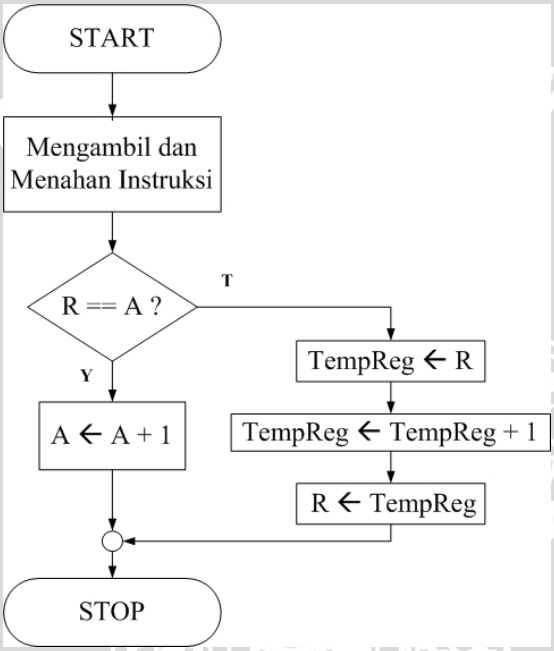
Tabel 6 Instruksi SBB dan Op-code

Instruksi	Opcode
SBB A	9F
SBB B	98
SBB C	99
SBB D	9A
SBB E	9B
SBB H	9C
SBB L	9D

4.2.2.5 Instruksi INR

Instruksi INR adalah instruksi yang memerintahkan mikroprosesor untuk menambahkan suatu register dengan angka satu. Selain status flag carry,nilai dalam status flag akan berubah untuk merefleksikan hasil penambahan angka satu dalam instruksi ini.

Dalam sistem yang dirancang ini, INR digunakan untuk menambahkan angka satu dalam register saja sehingga penulisan instruksi ini adalah INR R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi INR ditunjukkan dalam Gambar 4.13.



Gambar 4.13 Diagram Alir Instruksi INR

Jika dijabarkan dalam step-step, instruksi INR dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Jika instruksi mengacu pada register akumulator, maka akumulator langsung ditambahkan satu, jika tidak maka isi register sumber diduplikasi ke dalam register sementara
4) isi register sementara ditambah dengan angka satu kemudian isi register sementara diduplikasi ke register tempat data awal berasal
5) Instruksi berakhir

Instruksi INR dan op-code masing-masing instruksi dijabarkan dalam Tabel 7.

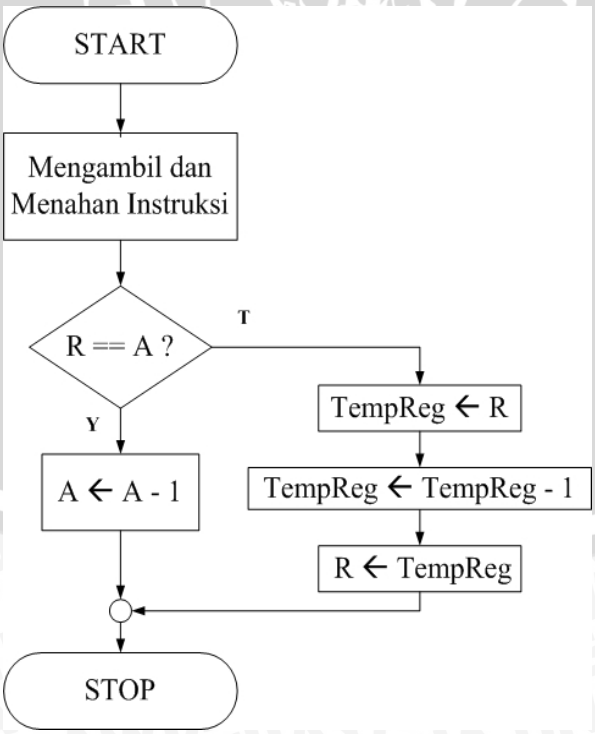
Tabel 7 Instruksi INR dan Op-code

Instruksi	Opcode
INR A	3C
INR B	04
INR C	0C
INR D	14
INR E	1C
INR H	24
INR L	2C

4.2.2.6 Instruksi DCR

Instruksi DCR adalah instruksi yang memerintahkan mikroprosesor untuk mengurangi suatu register dengan angka satu. Selain status flag,nilai dalam status flag akan berubah untuk merefleksikan hasil operasi dalam instruksi ini.

Dalam sistem yang dirancang ini, DCR digunakan untuk mengurangkan suatu angka dengan angka satu dalam register saja sehingga penulisan instruksi ini adalah DCR R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi DCR ditunjukkan dalam Gambar 4.14.



Gambar 4.14 Diagram Alir Instruksi DCR

Jika dijabarkan dalam step-step, instruksi DCR dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Jika instruksi mengacu pada register akumulator, maka akumulator langsung dikurangi satu, jika tidak maka isi register sumber diduplikasi ke dalam register sementara
4) isi register sementara dikurangi dengan angka satu kemudian isi register sementara diduplikasi ke register tempat data awal berasal
5) Instruksi berakhir

Instruksi DCR dan op-code masing-masing instruksi dijabarkan dalam Tabel 8.

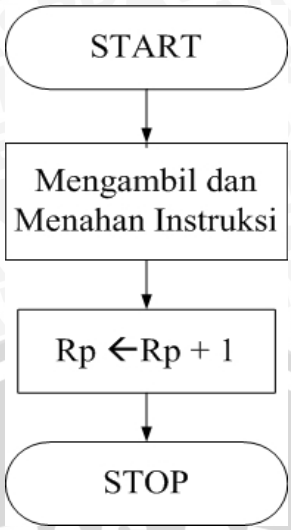
Tabel 8 Instruksi DCR dan Op-code

Instruksi	Opcode
DCR A	3D
DCR B	05
DCR C	0D
DCR D	15
DCR E	1D
DCR H	25
DCR L	2D

4.2.2.7 Instruksi INX

Instruksi INX adalah instruksi yang memerintahkan mikroprosesor untuk menambahkan suatu register dengan angka satu. Instruksi ini hampir sama dengan instruksi INR namun instruksi INX digunakan untuk register pair yaitu register BC, DE, dan HL. Operasi ini tidak akan mengubah status flag.

Dalam sistem yang dirancang ini, INX digunakan untuk menambahkan angka satu dalam register pair saja sehingga penulisan instruksi ini adalah INX Rp. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 6 T states. Diagram alir untuk instruksi INX ditunjukkan dalam Gambar 4.15.



Gambar 4.15 Diagram Alir Instruksi INX

Jika dijabarkan dalam step-step, instruksi INX dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Register pair ditambahkan dengan angka satu
4) Instruksi berakhir

Instruksi INX dan op-code masing-masing instruksi dijabarkan dalam Tabel 9.

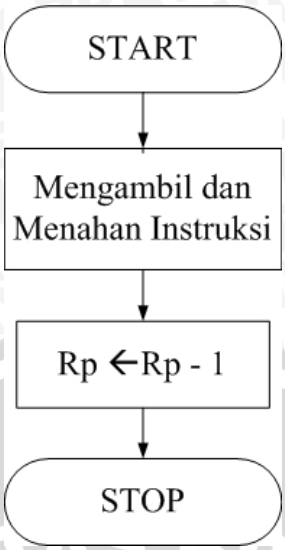
Tabel 9 Instruksi INX dan Op-code

Instruksi	Opcode
INX B	03
INX D	13
INX H	23

4.2.2.8 Instruksi DCX

Instruksi DCX adalah instruksi yang memerintahkan mikroprosesor untuk menambahkan suatu register dengan angka satu. Instruksi ini hampir sama dengan instruksi DCR namun instruksi DCX digunakan untuk register pair yaitu register BC, DE, dan HL. Operasi ini tidak akan mengubah status flag.

Dalam sistem yang dirancang ini, DCX digunakan untuk mengurangi suatu angka satu dalam register pair saja sehingga penulisan instruksi ini adalah DCX R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 6T states. Diagram alir untuk instruksi DCX ditunjukkan dalam Gambar 4.16.



Gambar 4.16 Diagram Alir Instruksi DCX

Jika dijabarkan dalam step-step, instruksi DCX dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Register pair dikurangi dengan angka satu
4) Instruksi berakhir

Instruksi DCX dan op-code masing-masing instruksi dijabarkan dalam Tabel 10.

Tabel 10 Instruksi DCX dan Op-code

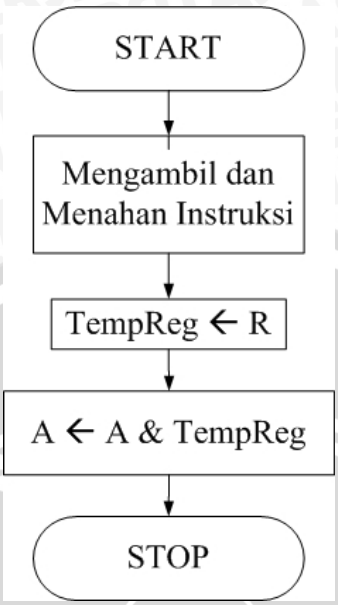
Instruksi	Opcode
DCX B	08
DCX D	18
DCX H	28

4.2.2.9 Instruksi ANA

Instruksi ANA adalah instruksi yang memerintahkan mikroprosesor untuk meng-AND-kan isi suatu register dengan register akumulator kemudian hasilnya diletakkan ke dalam register akumulator. Nilai dalam status flag akan berubah untuk merefleksikan hasil operasi dalam instruksi ini dan untuk status flag carry akan direset atau bernilai 0.

Dalam sistem yang dirancang ini, ANA digunakan untuk meng-AND-kan isi akumulator dengan register saja sehingga penulisan instruksi ini adalah ANA R.

Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi ANA ditunjukkan dalam Gambar 4.17.



Gambar 4.17 Diagram Alir Instruksi ANA

Jika dijabarkan dalam step-step, instruksi ANA dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan Instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register sementara di-AND-kan dengan isi register akumulator,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi ANA dan op-code masing-masing instruksi dijabarkan dalam Tabel 11.

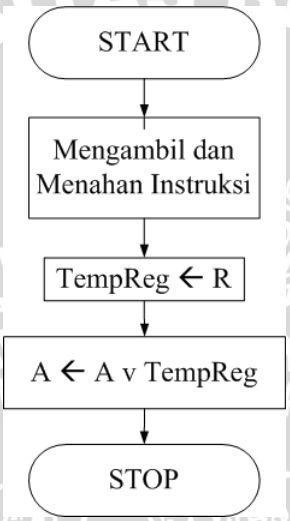
Tabel 11 Instruksi ANA dan Op-code

Instruksi	Opcode
ANA A	A7
ANA B	A0
ANA C	A1
ANA D	A2
ANA E	A3
ANA H	A4
ANA L	A5

4.2.2.10 Instruksi ORA

Instruksi ORA adalah instruksi yang memerintahkan mikroprosesor untuk meng-OR-kan isi suatu register dengan register akumulator kemudian hasilnya diletakkan ke dalam register akumulator. Nilai dalam status flag akan berubah untuk merefleksikan hasil operasi dalam instruksi ini dan untuk status flag carry dan auxiliary carry akan direset atau bernilai nol.

Dalam sistem yang dirancang ini, ORA digunakan untuk meng-OR-kan isi akumulator dengan register saja sehingga penulisan instruksi ini adalah ORA R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi ORA ditunjukkan dalam Gambar 4.18.



Gambar 4.18 Diagram Alir Instruksi ORA

Jika dijabarkan dalam step-step, instruksi ORA dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register sementara di-OR-kan dengan isi register akumulator,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi ORA dan op-code masing-masing instruksi dijabarkan dalam Tabel 12.

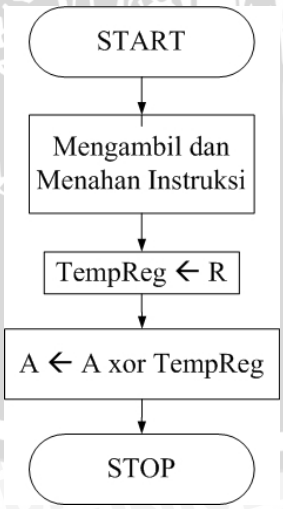
Tabel 12 Instruksi ORA dan Op-code

Instruksi	Opcode
ORA A	B7
ORA B	B0
ORA C	B1
ORA D	B2
ORA E	B3
ORA H	B4
ORA L	B5

4.2.2.11 Instruksi XRA

Instruksi XRA adalah instruksi yang memerintahkan mikroprosesor untuk meng-XOR-kan isi suatu register dengan register akumulator kemudian hasilnya diletakkan ke dalam register akumulator. Nilai dalam status flag akan berubah untuk merefleksikan hasil operasi dalam instruksi ini dan untuk status flag carry dan auxiliary carry akan direset atau bernilai nol.

Dalam sistem yang dirancang ini, XRA digunakan untuk meng-XOR-kan isi akumulator dengan register saja sehingga penulisan instruksi ini adalah XRA R. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4 T states. Diagram alir untuk instruksi XRA ditunjukkan dalam Gambar 4.19.



Gambar 4.19 Diagram Alir Instruksi XRA

Jika dijabarkan dalam step-step, instruksi XRA dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register sumber diduplikasi ke dalam register sementara
4) Isi register sementara di-XOR-kan dengan isi register akumulator,hasilnya diletakkan dalam register akumulator
5) Instruksi berakhir

Instruksi XRA dan op-code masing-masing instruksi dijabarkan dalam Tabel 13.

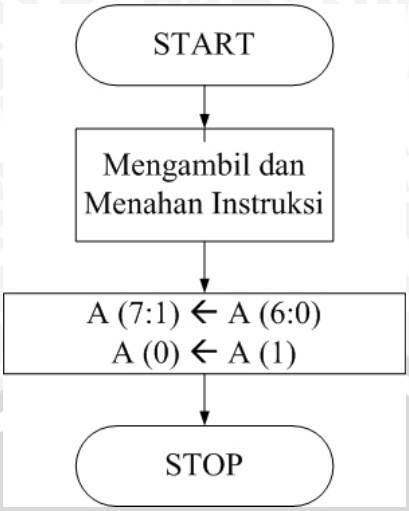
Tabel 13 Instruksi XRA dan Op-code

Instruksi	Opcode
XRA A	AF
XRA B	A8
XRA C	A9
XRA D	AA
XRA E	AB
XRA H	AC
XRA L	AD

4.2.2.12 Instruksi RLC

Instruksi RLC adalah instruksi yang memerintahkan mikroprosesor untuk menggeser angka dalam register akumulator 1 bit ke kiri tanpa melalui carry. Hanya nilai status flag carry yang berubah saat operasi ini dijalankan.

. Opcode instruksi RLC adalah 07 dengan penulisan instruksi RLC. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4T states. Diagram alir untuk instruksi RLC ditunjukkan dalam Gambar 4.20.



Gambar 4.20 Diagram Alir Instruksi RLC

Jika dijabarkan dalam step-step, instruksi RLC dapat dijabarkan sebagai berikut.

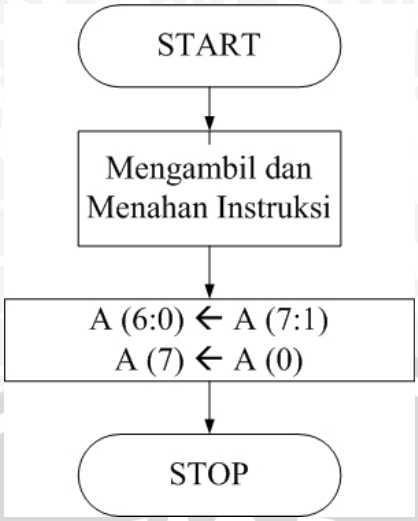
1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register akumulator digeser ke kiri sebanyak 1 bit
4) Instruksi berakhir

Opcode instruksi RLC adalah 07 dengan penulisan instruksi RLC.

4.2.2.13 Instruksi RRC

Instruksi RRC adalah instruksi yang memerintahkan mikroprosesor untuk menggeser angka dalam register akumulator 1 bit ke kanan tanpa melalui carry. Hanya nilai status flag carry yang berubah saat operasi ini dijalankan.

Opcode instruksi RRC adalah 0F dengan penulisan instruksi RRC. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4T states. Diagram alir untuk instruksi RRC ditunjukkan dalam Gambar 4.21.



Gambar 4.21 Diagram Alir Instruksi RRC

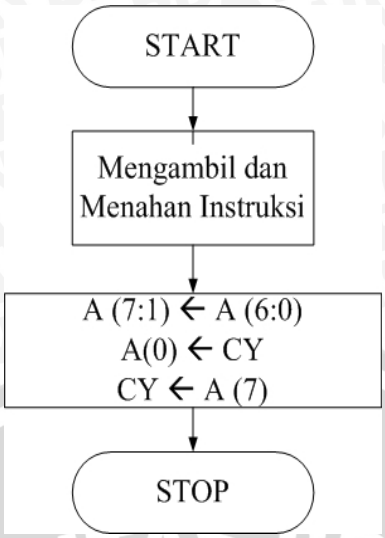
Jika dijabarkan dalam step-step, instruksi RRC dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register akumulator digeser ke kanan sebanyak 1 bit
4) Instruksi berakhir

4.2.2.14 Instruksi RAL

Instruksi RAL adalah instruksi yang memerintahkan mikroprosesor untuk menggeser angka dalam register akumulator 1 bit ke kiri melalui carry. Hanya nilai status flag carry yang berubah saat operasi ini dijalankan.

Opcode instruksi RAL adalah 17 dengan penulisan instruksi RAL. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4T states. Diagram alir untuk instruksi RAL ditunjukkan dalam Gambar 4.22.



Gambar 4.22 Diagram Alir Instruksi RAL

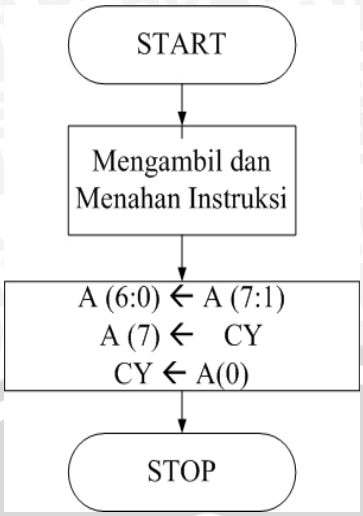
Jika dijabarkan dalam step-step, instruksi RAL dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register akumulator digeser ke kiri sebanyak 1 bit melalui carry
4) Instruksi berakhir

4.2.2.15 Instruksi RAR

Instruksi RAR adalah instruksi yang memerintahkan mikroprosesor untuk menggeser angka dalam register akumulator 1 bit ke kanan melalui carry. Hanya nilai status flag carry yang berubah saat operasi ini dijalankan.

Opcode instruksi RAR adalah 1F dengan penulisan instruksi RAR. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4T states. Diagram alir untuk instruksi RAR ditunjukkan dalam Gambar 4.23.



Gambar 4.23 Diagram Alir Instruksi RAR

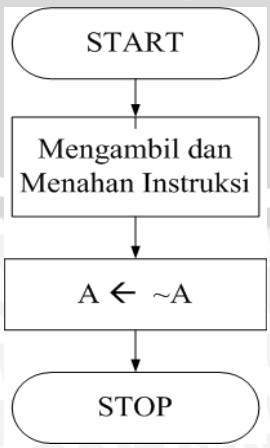
Jika dijabarkan dalam step-step, instruksi RAR dapat dijabarkan sebagai berikut.

1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register akumulator digeser ke kanan sebanyak 1 bit melalui carry
4) Instruksi berakhir

4.2.2.16 Instruksi CMA

Instruksi CMA adalah instruksi yang memerintahkan mikroprosesor untuk menegasikan angka dalam register akumulator. Tidak ada nilai status flag yang berubah karena operasi ini.

Opcode instruksi CMA adalah 2F dengan penulisan instruksi CMA. Instruksi ini adalah instruksi 1 byte yang memerlukan 1 kali siklus mesin dengan 4T states. Diagram alir untuk instruksi CMA ditunjukkan dalam Gambar 4.24.



Gambar 4.24 Diagram Alir Instruksi CMA

Jika dijabarkan dalam step-step, instruksi CMA dapat dijabarkan sebagai berikut.

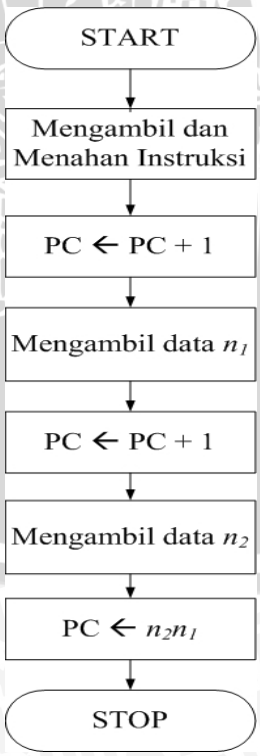
1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Isi register akumulator dinegasikan kemudian diletakkan dalam register akumulator kembali
4) Instruksi berakhir

4.2.3 Instruksi Percabangan

Instruksi percabangan adalah instruksi yang menyebabkan program counter menuju ke alamat percabangan tanpa menyebabkan perubahan isi register lain. Pada perancangan ini hanya dirancang untuk dapat melakukan percabangan tanpa syarat (unconditional) yaitu instuksi JMP.

Instruksi JMP adalah instruksi yang menyebabkan program counter ke suatu alamat percabangan yang ditentukan tanpa syarat. Tidak ada nilai status flag yang berubah saat operasi ini dijalankan.

Opcode instruksi JMP adalah C3 dengan penulisan instruksi JMP. Instruksi ini adalah instruksi 3 byte yang memerlukan 3 kali siklus mesin dengan 7T states. Diagram alir untuk instruksi JMP ditunjukkan dalam Gambar 4.25.



Gambar 4.25 Diagram Alir Instruksi JMP

Jika dijabarkan dalam step-step, instruksi JMP dapat dijabarkan sebagai berikut.

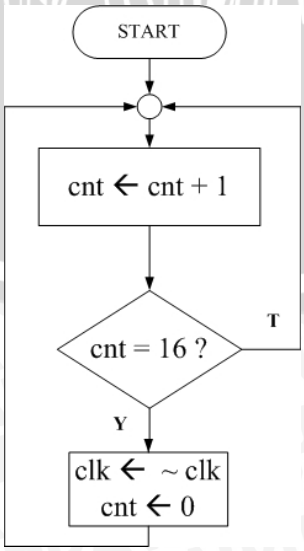
1) Pengambilan instruksi dari memori
2) Penerjemahan instruksi
3) Pengambilan data atau operand sebanyak 2 kali
4) Program counter diset sesuai dengan operand
5) Instruksi berakhir

4.3 Perancangan Tiap Blok Penyusun Sistem

Berdasar penjelasan gambaran kerja sistem serta instruksi-instruksi yang dapat dijelaskan oleh sistem maka dilakukan pembuatan blok bagian penyusun sistem menggunakan bahasa VHDL dan software yang digunkana adalah Xilinx ISE 11.1. Unit-unit tersebut antara lain sistem clock, ALU dan status flag, buffer, latch, dekoder, input output, dan memori. Seluruh unit dalam sistem akan bekerja sama satu dengan yang lain dalam mengeksekusi perintah masukan user sehingga sistem dapat bekerja sesuai dengan yang diinginkan.

4.3.1 Unit Pembagi Clock

Unit pembagi clock digunakan untuk mengatur clock masukan sebesar 50 MHz (didapatkan dari Kristal yang telah tersedia pada FPGA) menjadi 3,25 MHz. Frekuensi 3,25 MHz adalah frekuensi clock maksimal mikroprosesor 8085 sesuai dengan datasheet mikroprosesor 8085. Unit ini memiliki input *clock* (clock fpga) dan ouput *clock_8085* dan *clock_fpga* (sebagai *display* agar hasil input clock terlihat). Blok ini memiliki *variable cnt* yang berfungsi sebagai penghitung clock mula-mula.Diagram alir untuk unit sistem clock ditunjukkan dalam Gambar 4.26.



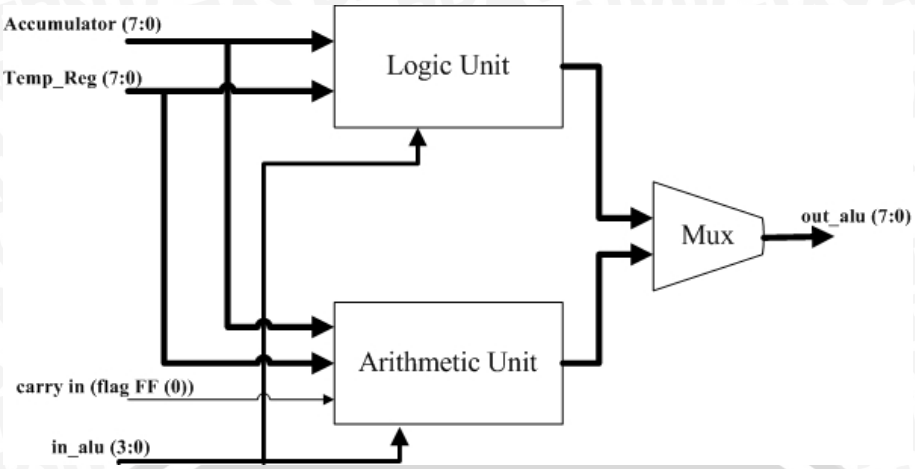
Gambar 4.26 Diagram Alir Unit Sistem *Clock*

Berdasarkan algoritma yang telah dirancang maka disusunlah listing program sebagai berikut:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity cekclock is
    Port ( clock : in  STD_LOGIC;
           clock8085 : out  STD_LOGIC;
           clock_fpga : out  STD_LOGIC);
end cekclock;

architecture Behavioral of cekclock is
begin
clock_fpga <= clock;
process (clock)
variable cnt : integer range 0 to 16;
begin
if (clock'event and clock='1') then
if(cnt=16)then
cnt:=0;
clock8085<='1';
elsif (cnt=8) then
clock8085 <= '0';
cnt := cnt+1;
else
cnt := cnt+1;
end if;
end if;
end process;
end Behavioral;
```

4.3.2 Unit ALU dan Status Flag

Sesuai dengan namanya, ALU (*Arithmetic Logic Unit*) mampu untuk mengeksekusi dua macam operasi, yaitu operasi aritmatika dan logika. Dalam perancangan ini, ALU disusun atas unit aritmatika, unit, logika, dan unit multiplekser. Hal ini dilakukan agar pemrosesan aritmatika dan logika dapat berjalan dengan baik sesuai dengan fungsi masing-masing tanpa mencampurkan proses aritmatika dan logika tersebut. Diagram blok rancangan blok ALU digambarkan dalam Gambar 4.27.



Gambar 4.27 Diagram Blok Rancangan ALU

Sesuai dengan sistem mikroprosesor 8085, input ALU adalah data pada register akumulator dan register sementara (temporary register) yang masing-masing data berjumlah 8 bit. Unit aritmatika adalah blok unit pemroses aritmatika berbasis full adder. Oleh karena itu, input unit aritmatika berjumlah 3, yaitu data pada akumulator, data pada register sementara, dan data carry. Carry didapat dari flip-flop flag bit ke 0. Data flip-flop flag didapat dari status-status hasil operasi ALU. Data register sementara didapat dari register B,C,D,E,H, atau L.Out_alu adalah hasil operasi ALU yang nantinya akan dimasukkan ke dalam register-register yang terdapat dalam mikroprosesor 8085. Register yang akan disimpani hasil tersebut bergantung dari terjemahan instruksi yang diterjemahkan dalam dekoder instruksi. Dalam gambar diagram blok rancangan ALU, terdapat input yg bernama in_alu. In_alu disini berfungsi sebagai selektor, yaitu menyeleksi blok unit yang mana yang akan memproses input. In_alu berjumlah 5 bit yang diatur berdasarkan jumlah operasi yang dapat dilakukan oleh blok ALU tersebut. Operasi yang dapat dilakukan oleh ALU atau dinamakan daftar kebenaran ALU dicantumkan dalam Tabel 14.

Tabel 14 Daftar Kebenaran Unit ALU

in_alu	**Operasi**	**Fungsi**
00000	out_alu <= A - Temp_Reg	Pengurangan A dengan Temp_Reg
00001	out_alu <= A+1	Increment A
00010	out_alu <= A-1	Decrement A
00011	out_alu <= A – TempReg - Cy_{in}	Pengurangan A, Temp_Reg dan Cy_{in}
00100	out_alu <= Temp_Reg +1	Increment Temp_Reg

00101	out_alu <= Temp_Reg -1	Decrement Temp_Reg
00110	out_alu <= A+Temp_Reg	Penjumlahan A dan Temp_Reg
00111	out_alu <= A+Temp_Reg+ Cy_{in}	Penjumlahan A dan Temp_Reg dengan carry in dari flip-flop flag
10000	out_alu <= NOT A	Complement A
10001	out_alu <= NOT Temp_Reg	Complement Temp_Reg
10010	out_Alu <= A AND Temp_Reg	AND
10011	out_alu <= A OR Temp_Reg	OR
10100	out_alu <= A XOR Temp_Reg	XOR
10101	out_alu <= A ←	Menggeser 1 bit A ke kiri
10110	out_alu <= → A	Menggeser 1 bit A ke kanan
11000	out_alu<= (Cy_{in},A) ←	Menggeser 1 bit A ke kiri melalui carry
11001	out_alu<= → (Cy_{in},A)	Menggeser 1 bit A ke kanan melalui carry

Prinsip status flag adalah memberi tanda khusus pada tiap-tiap flag bergantung instruksi yang diperintahkan user dan hasil operasi ALU. Pada rancangan sistem ini, selain status carry dan auxiliary carry, status flag didapatkan dari pendeteksian hasil output ALU 8 bit. Untuk status carry dan auxiliary carry didapat dari pengecekan overflow suatu hasil operasi. Daftar kebenaran status flag ditunjukkan dalam Tabel 15.

Tabel 15 Daftar Kebenaran Status Flag

Nama Status flag	Kondisi hasil operasi ALU	Kondisi flag
Carry	Overflow (terdapat carry)	1
	No overflow (tidak terdapat carry)	0
Parity	Bit bernilai 1 berjumlah genap	1
	Bit bernilai 1 berjumlah ganjil	0
Auxiliarry Carry	Overflow dari bit ke-3 ke bit ke-4	1
	No overflow dari bit ke-3 ke bit ke-4	0
Zero	Hasil ALU = 0	1
	Hasil ALU tidak sama dengan 0	0
Sign	Bit ke 7 ALU bernilai 1 (negatif)	1
	Bit ke 7 ALU bernilai 0 (positif)	0

Berdasarkan diagram blok dan penjelasan rancangan yang telah dikemukakan maka disusunlah listing program proses unit ALU sebagai berikut.

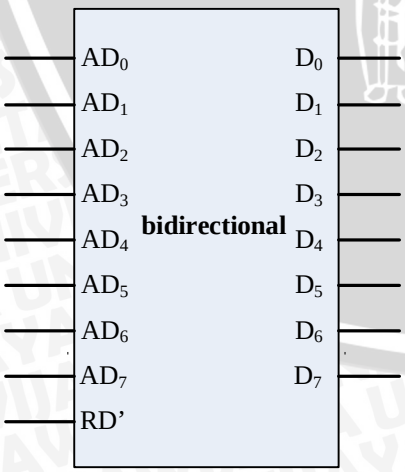
```
process
(RegA,TempReg,carry_in,in_alu,arith,logic,out_alu,arith2,flag_reg,flag
_check)
begin
carry_in <= flag_check(0);
case in_alu (3 downto 0) is
when "0000" => arith <= ('0' & RegA) - ('0' & TempReg);
               logic <= NOT RegA;
               arith2 <= ('0' & RegA (3 downto 0)) –
                         ('0' & TempReg (3 downto 0));
when "0001" => arith <= ('0' & RegA) + "000000001";
               logic <= NOT TempReg;
               arith2 <= ('0' & RegA (3 downto 0)) + "00001";
when "0010" => arith <= ('0' & RegA) - "000000001";
               logic <= RegA AND TempReg;
               arith2 <= ('0' & RegA (3 downto 0)) - "00001";
when "0011" => arith <= ('0' & RegA)-('0' & TempReg) - carry_in;
               logic <= RegA OR TempReg;
               arith2 <= ('0' & RegA (3 downto 0))-
               ('0' & TempReg (3 downto 0)) - carry_in;
when "0100" => arith <= ('0' & TempReg) + "000000001";
               logic <= RegA XOR TempReg;
               arith2 <= ('0' & TempReg (3 downto 0)) + "00001";
when "0101" => arith <= ('0' & TempReg) - "000000001";
               arith2 <= ('0' & TempReg (3 downto 0)) - "00001";
               logic (7 downto 1) <= RegA (6 downto 0);
               logic(0) <= RegA (7);
when "0110" => arith <= ('0' & RegA) +('0' & TempReg);
               arith2 <= ('0' & RegA (3 downto 0)) +
               ('0' & TempReg (3 downto 0));
               logic (6 downto 0) <= RegA (7 downto 1);
               logic (7) <= RegA (0);
when "0111" => arith <= ('0' & RegA)+('0' & TempReg) + carry_in;
               arith2 <= ('0' & RegA (3 downto 0))+
               ('0' & TempReg (3 downto 0)) + carry_in;
               logic <= out_alu;
when "1000" => logic (7 downto 1) <= RegA (6 downto 0);
               carry_out <= RegA (7);
               logic (0) <= carry_in;
               arith <= arith;
when "1001" => logic (6 downto 0) <= RegA (7 downto 1);
               carry_out <= RegA (0);
               logic (7) <= carry_in;
               arith <= arith;

-- for INX
when "1010" => arith <= ('0' & TempReg) + carry_in;
when others => arith <= arith; logic <= logic;
               carry_out <= carry_out;
end case;
if (in_alu(4)='0') then out_alu <= arith (7 downto 0);
                        carry_out <= arith(8);
else out_alu <= logic;
end if;

aux_carry <= arith2(4);
parity <= ((out_alu(0) XOR out_alu(1)) XOR (out_alu(2)
          XOR out_alu(3)))
```

```
            XOR  ((out_alu(4)  XOR  out_alu(5))  XOR  (out_alu(6)  XOR
            out_alu(7)));
end process;

process (flag_reg,parity,out_alu,carry_out,aux_carry,in_alu)
begin
flag_reg(0)<= carry_out;

if (parity='0') then flag_reg(2) <= '1';
elsif (parity='1') then flag_reg(2) <= '0';
end if;
flag_reg(4) <= aux_carry;
if (out_alu="00000000") then flag_reg(6) <= '1';
      else flag_reg(6) <= '0';
end if;
flag_reg(7) <= out_alu (7);
end process;
```

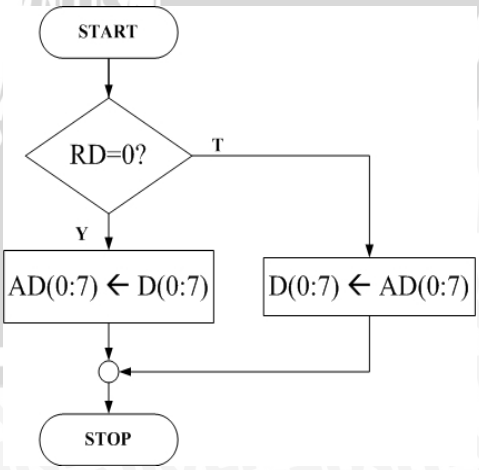
4.3.3 Unit Bidirectional, Buffer, dan Dekoder

Unit bidirectional, buffer, dan dekoder adalah unit yang menjembatani antara mikroprosesor dengan unit input output ataupun memori. Unit bidirectional dalam sistem berfungsi untuk mengendalikan kapan saatnya data masuk atau keluar sehingga dapat bersifat bidirectional. Kontrol unit bidirectional didapatkan dari keluaran pin RD'. Unit buffer berfungsi untuk mengendalikan kapan saatnya alamat terkirim pada unit yang dikehendaki untuk diakses. Unit buffer dikendalikan oleh pin ALE.

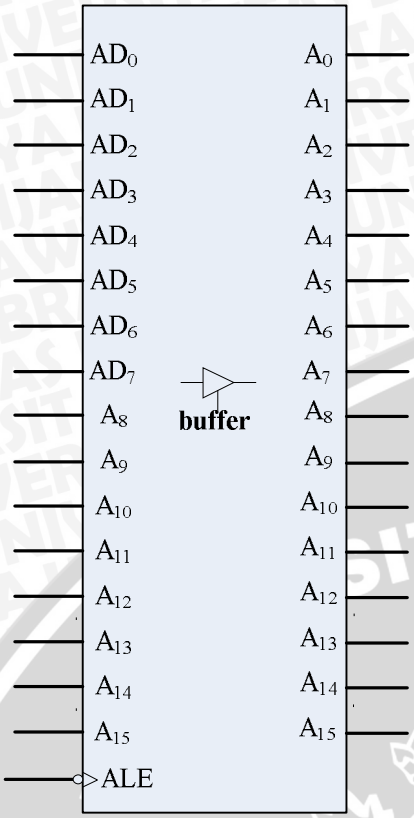
Masukan unit dekoder terdiri atas IO/M, RD', dan WR' yang menghasilkan empat keluaran yaitu MEMR',MEMW', IOR', dan IOW'. Hasil keluaran dekoder inilah yang akan menentukan akses baca atau tulis dari atau ke unit input output maupun ke unit memori. Diagram blok dan diagram alir unit bidirectional, buffer, dan dekoder ditunjukkan dalam Gambar 4.28 (a) sampai Gambar 4.28 (f).



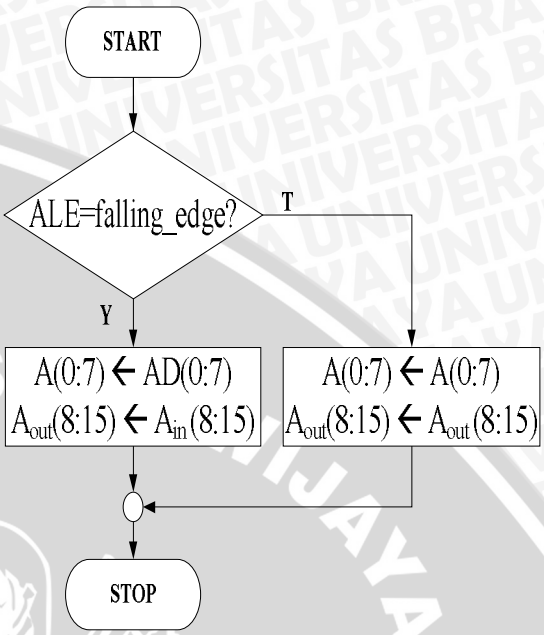
(a)



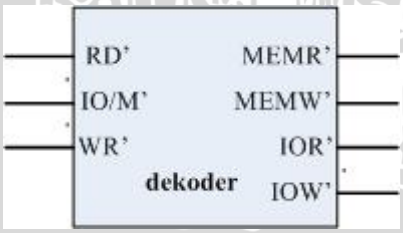
(b)


AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A8
A9
A10
A11
A12
A13
A14
A15
ALE
buffer
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15

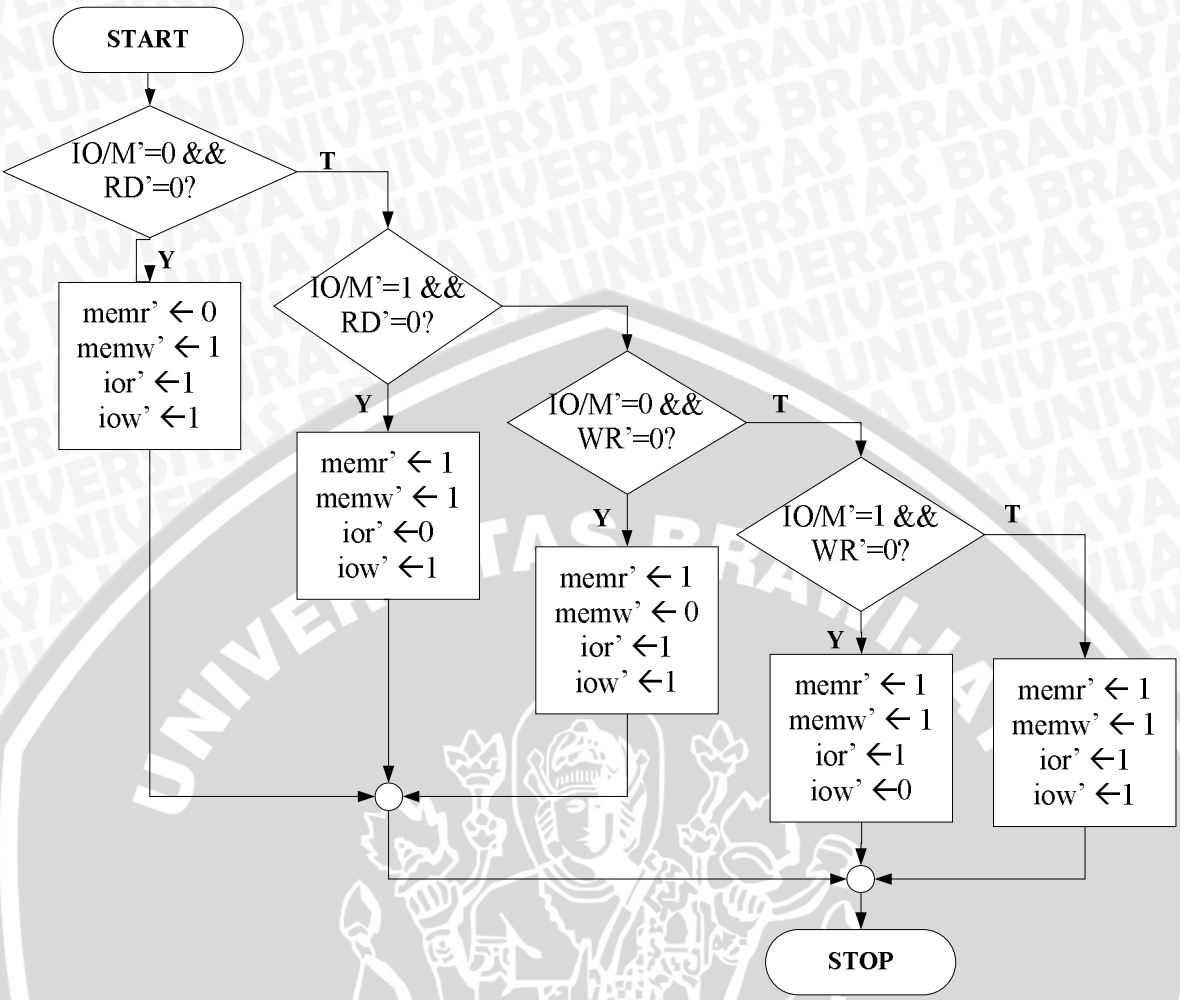
(c)


START
ALE=falling_edge?
T
Y
A(0:7) ← AD(0:7)
Aout(8:15) ← Ain(8:15)
A(0:7) ← A(0:7)
Aout(8:15) ← Aout(8:15)
STOP

(d)


RD'
IO/M'
WR'
dekoder
MEMR'
MEMW'
IOR'
IOW'

(e)



(f)

Gambar 4.28 (a) Diagram Blok Bidirectional

(b) Diagram Alir Bidirectional

(c) Diagram Blok Buffer

(d) Diagram Alir Buffer

(e) Diagram Blok Dekoder

(f) Diagram Alir Dekoder

Berdasarkan diagram blok dan penjelasan rancangan yang telah dikemukakan maka disusunlah listing program proses unit-unit tersebut sebagai berikut.

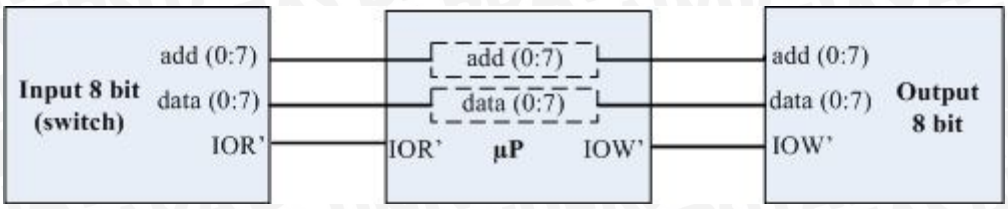
```
process (datain,dataout,INbidirect,OUTbidirect,
inIO,alamatINPUT,alamatOUTPUT,outMEM,inMEM,RD,ALE,alamatmembwh)
begin
case RD is
when '1' => if falling_edge (ALE) then alamatoutput <= dataout;
                                       alamatinput <= dataout;
                                       alamatmembwh <= dataout;
                                       alamatmematas <= alamathigh;
```

```
                                    end if;
                            outMEM <= dataout;
                             outIO <= dataout;
when others => inbidirect <= datain;
end case;
end process;

process (memr,ior,datain,inMEM,inIO)
begin
if (memr='0') then datain <= inMEM;
elsif (ior='0') then datain <= inIO;

end if;
end process;

process (IO_M,RD,WR)
begin
if (IO_M='0' and RD='0' and WR='1') then memr <= '0';ior <= '1';
                                        memw <= '1'; iow <= '1';
elsif (IO_M='1' and RD='0' and WR='1') then  ior <= '0';memr <= '1';
                                             memw <= '1';
                                             iow <= '1';
elsif (IO_M='0' and WR='0' and RD='1') then  memw <= '0';memr <= '1';
                                            ior <= '1'; iow <= '1';
elsif (IO_M='1' and WR='0' and RD='1') then  iow <= '0';memr <= '1';
                                             ior <= '1';
                                             memw <= '1';
else iow <= '1'; ior <= '1'; memw <= '1'; memr <= '1';
end if;
end process;
```

4.3.4 Unit Input dan Unit Output

Unit input adalah unit yang berfungsi menerima masukan user dan menyalurkan data ke dalam mikroprosesor saat dibutuhkan. Mikroprosesor memiliki kemampuan untuk mengakses unit input sesuai dengan alamat unit input masing-masing. Unit input yang dirancang dalam sistem ini adalah unit input yang memiliki alamat 12H sehingga data akan terbaca oleh mikroprosesor jika mikroprosesor mengirimkan alamat 12H dan IOR' aktif.

Unit output adalah unit yang berfungsi menampilkan hasil keluaran saat mikroprosesor menginstruksikan untuk menulis hasil keluaran pada port output. Unit output yang dirancang dalam sistem ini adalah unit output yang memiliki alamat 13H sehingga data akan terbaca oleh mikroprosesor jika mikroprosesor mengirimkan alamat 13H dan IOW' aktif. Diagram blok rancangan unit input dan output ditunjukkan dalam Gambar 4.29.



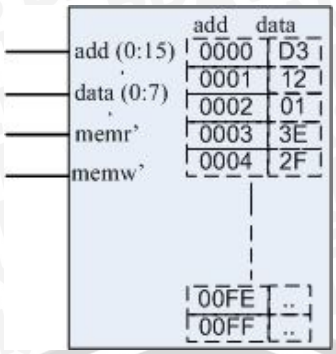
Gambar 4.29 Diagram Blok Unit Input dan Output

Berdasarkan diagram blok dan penjelasan rancangan yang telah dikemukakan maka disusunlah listing program input output tersebut sebagai berikut.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity inputoutput is
    Port ( input_luar : in  STD_LOGIC_VECTOR (7 downto 0);
           output : out  STD_LOGIC_VECTOR (7 downto 0);
           IOR : in  STD_LOGIC;
           act_addr : in  STD_LOGIC;
           IOW : in  STD_LOGIC);
end inputoutput;

architecture Behavioral of inputoutput is
signal data,alamat : STD_LOGIC_VECTOR (7 downto 0):=(others=>'0');

begin
process (act_addr,alamat,IOR,IOW,input_luar,data)
begin
if (act_addr='1') then alamat <= input_luar;
elsif (alamat="00010010" and IOR='1') then data <= input_luar;
elsif (alamat="00010011" and IOW='1') then output <= data;
else data <= data;alamat <= alamat;
end if;
end process;
end Behavioral;
```

4.3.5 Unit Memori

Unit memori adalah unit yang berfungsi menampung semua instruksi yang dimasukkan user dan berfungi sebagai sumber materi yang akan dikerjakan mikroprosesor. Unit memori yang digunakan adalah unit memori bertipe RAM (*random access memory*) dengan jalur data 8 bit dan jalur alamat 16 bit. RAM yang dirancang memiliki kapasitas 255 byte. Diagram blok rancangan unit input dan output ditunjukkan dalam Gambar 4.30.



Gambar 4.30 Diagram Blok Unit Memori

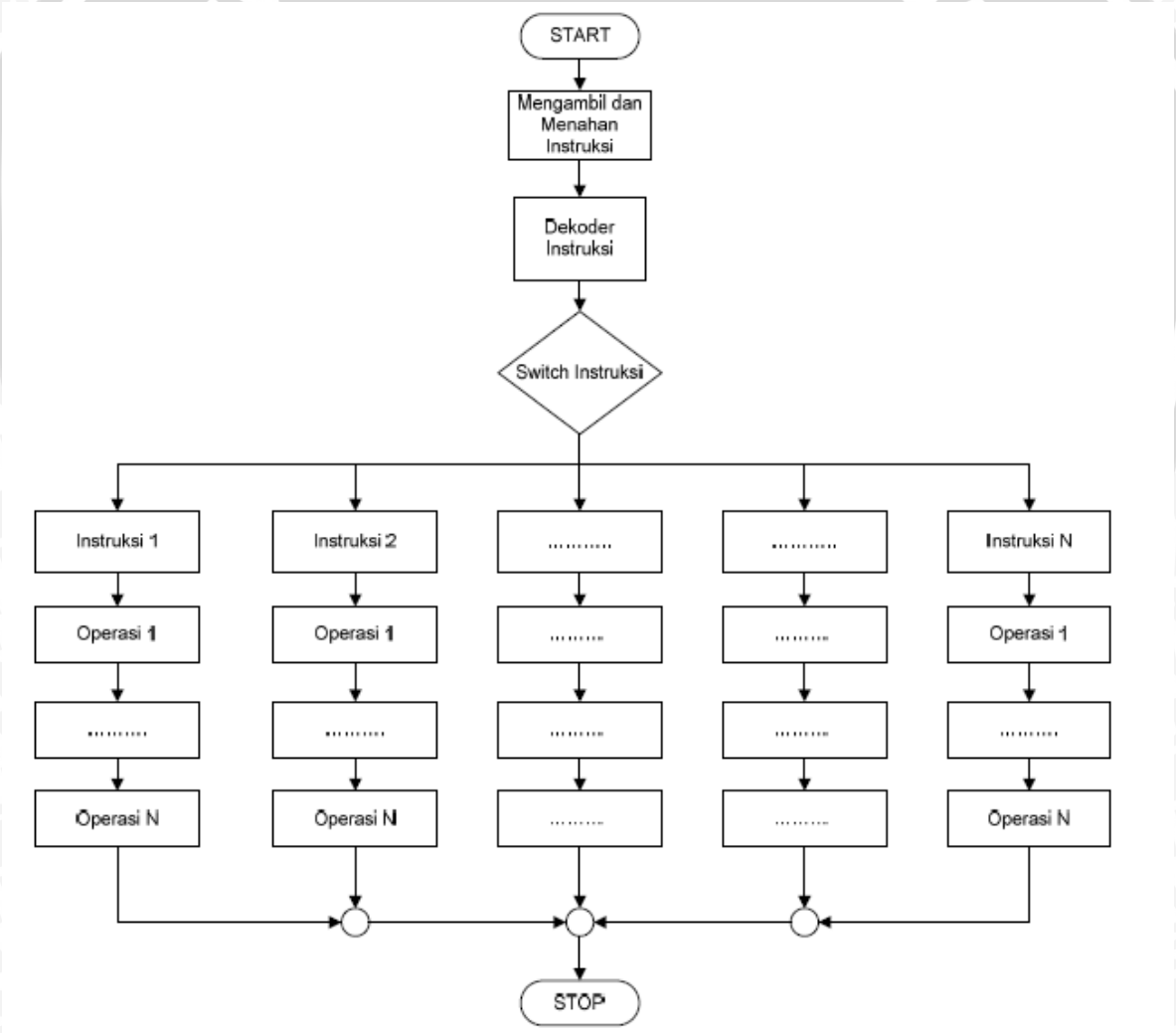
Berdasarkan diagram blok dan penjelasan rancangan yang telah dikemukakan maka disusunlah listing program unit memori tersebut sebagai berikut.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity rambaru is
 Port ( CLOCK : in STD_LOGIC;
                  memr : in  STD_LOGIC;
                  memw : in  STD_LOGIC;
                  ledset: out  STD_LOGIC_VECTOR (2 downto 0);
                  data_out : out  STD_LOGIC_VECTOR (7 downto 0);
                  input_luar : in STD_LOGIC_VECTOR (7 downto 0) ;
                  setinput: in STD_LOGIC_VECTOR (2 downto 0)
                 );
end rambaru;
architecture Behavioral of rambaru is
TYPE RAM IS ARRAY(0 TO 255) OF std_logic_vector(7 DOWNTO 0);
SIGNAL ram_block : RAM;

signal alamat_DEKODER,alamat_MEM,data_in:
       STD_LOGIC_VECTOR (7 downto 0) := (others=>'0');
BEGIN
ledset <= setinput;

      PROCESS (setinput)
      BEGIN
            IF (setinput="001") then ALAMAT_MEM <= input_luar;
            ELSIF (setinput="010") then ALAMAT_DEKODER <= input_luar;
            ELSIF (setinput="100") then data_in <= input_luar;
            ELSE alamat_dekoder <= alamat_dekoder;
            alamat_mem <=    alamat_mem;data_in <= data_in;
            END IF;
      END PROCESS;


      PROCESS (alamat_dekoder,memr,memw,data_in,ram_block)
      BEGIN
            IF (ALAMAT_DEKODER="00000000") then
            IF (memw = '1') THEN
            ram_block(conv_integer(unsigned(ALAMAT_MEM))) <= data_in;
            ELSIF(memr='1')THEN  data_out  <=  ram_block  (conv_integer
            (unsigned (ALAMAT_MEM)));
            END IF;
            END IF;
      END PROCESS;
```

4.3.6 Penyusunan Blok Digital Menjadi Sistem Terpadu

Blok digital yang telah dibuat masih merupakan blok otonom yang berdiri sendiri. Untuk membetuk kesatuan sistem antar blok menjadi sistem digital terpadu, maka blok-blok digital tersebut disusun menggunakan metode *schematic*. *Schematic symbol* dari blok-blok digital tipe *file."vhd"* yang telah diprogram ditambahkan sebagai *source* pembentuk tipe *file."sch"*. Penyusunan blok-blok digital mengacu pada rancangan skema sistem kerja antar blok yang telah dirancang pada tahap perancangan gambaran kerja sistem. Penyatu blok-blok tersebut adalah unit kontrol

Unit kontrol adalah sebuah unit yang berfungsi sebagai kontrol utama jalannya mikroprosesor. Unit kontrol ini akan mengatur unit-unit lainnya berdasarkan instruksi yang didapatkan dan akan bekerja berdasarkan clock yang diterima sistem. Diagram alir unit kontrol ditunjukkan dalam Gambar 4.31.



Gambar 4.31 Diagram Alir Unit Kontrol

4.4 Implementasi Sistem ke Dalam IC FPGA Xilinx Spartan 3E-500 FG320

Sistem yang telah dirancang selanjutnya diimplementasikan ke dalam IC FPGA Xilinx Spartan 3E-500 FG320. Tahap implementasi desain ke dalam arsitektur FPGA terdiri dari beberapa langkah yang dapat dijelaskan sebagai berikut:

4.4.1 Pengaturan Relasi antara I/O Sistem dengan Kode Pin FPGA

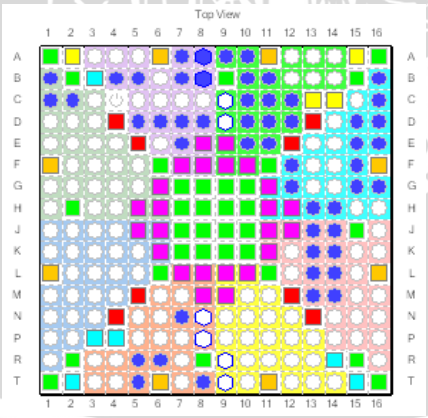
Tahap pertama yang perlu dilakukan adalah proses pengaturan relasi antara I/O sistem dengan kode pin FPGA. Pengaturan relasi antara I/O sistem dengan kode pin FPGA mengacu pada konfigurasi pin IC FPGA Xilinx 3E-500.

Relasi yang diperlukan adalah *fix input output, clock*, dan masukan 7-segment. Untuk mengatur relasi tersebut, diperlukan *file".ucf".File."ucf"* bentuk teks untuk relasi antara I/O sistem dengan konfigurasi pin FPGA ditunjukkan sebagai berikut:

```
NET "clock" LOC = B8;
NET "fixInputterpadu[0]"LOC = L15;
NET "fixInputterpadu[1]"LOC = K12;
NET "fixInputterpadu[2]"LOC = L17;
NET "fixInputterpadu[3]"LOC = M15;
NET "fixInputterpadu[4]"LOC = K13;
NET "fixInputterpadu[5]"LOC = L16;
NET "fixInputterpadu[6]" LOC = M14;
NET "fixInputterpadu[7]" LOC = M16;
NET "fixclock" LOC = B18;
NET "fixoutput[0]" LOC = J14;
NET "fixoutput[1]" LOC = J15;
NET "fixoutput[2]" LOC = K15;
NET "fixoutput[3]" LOC = K14;
NET "fixoutput[4]" LOC = E17;
NET "fixoutput[5]" LOC = P15;
NET "fixoutput[6]" LOC = F4;
NET "fixoutput[7]" LOC = R4;
NET "output13[0]" LOC = J12;
NET "output13[1]" LOC = G16;
NET "output13[2]" LOC = F14;
NET "output13[3]" LOC = H15;
NET "output13[4]" LOC = H16;
NET "output13[5]" LOC = G13;
NET "output13[6]" LOC = J16;
NET "output13[7]" LOC = G15;
NET "fixwrprog" LOC = E18;
```

```
NET "selectorOutput[0]" LOC = G18;
NET "selectorOutput[1]" LOC = H18;
NET "selectorOutput[2]" LOC = K18;
NET "selectorOutput[3]" LOC = K17;
NET "selectorinput[0]" LOC = L14;
NET "selectorinput[1]" LOC = L13;
NET "selectorinput[2]" LOC = N17;
NET "SevSeg[0]" LOC = L18;
NET "SevSeg[1]" LOC = F18;
NET "SevSeg[2]" LOC = D17;
NET "SevSeg[3]" LOC = D16;
NET "SevSeg[4]" LOC = G14;
NET "SevSeg[5]" LOC = J17;
NET "SevSeg[6]" LOC = H14;
NET "Anoda[0]" LOC = F17;
NET "Anoda[1]" LOC = H17;
NET "Anoda[2]" LOC = C18;
NET "Anoda[3]" LOC = F15;
NET "fixclock" CLOCK_DEDICATED_ROUTE = FALSE;
NET "STEPorRUN" LOC = R17;
NET "selectorOutput<1>" CLOCK_DEDICATED_ROUTE = FALSE;
```

Gambar konfigurasi pin-pin FPGA yang digunakan sebagai I/O sistem ditunjukkan dalam Gambar 4.32.



Gambar 4. 32 Konfigurasi Pin FPGA yang Digunakan Sebagai I/O Sistem

Pada gambar 4.32, pin yang diberi tanda warna biru tua adalah pin yang digunakan sebagai input/output dari sistem yang telah dirancang.

4.4.2 Synthesize

Proses *synthesize* berfungsi untuk mengkonversi desain dalam bentuk *VHDL Source Code* menjadi gerbang-gerbang logika yang dibutuhkan untuk menyusun keseluruhan sistem. Proses *synthesis* dilakukan oleh software *ISE Design Suite 11.1*. Melalui proses synthesis dihasilkan jumlah *slices*, *slice flip-flops*, 4 imput LUTs, dan GCLKs yang digunakan dalam menyusun sistem.

Jumlah *slices, slice flip-flops*, 4 input LUTs, dan GCLKs yang digunakan untuk menyusun sistem ditunjukkan dalam Tabel

Tabel 16 *Slices, slice flip-flops*, 4 input LUTs, dan BUFGMUXs Penyusun Sistem

Gerbang penyusun sistem	Gerbang Terpakai	Gerbang yang Tersedia	Prosentase Gerbang Terpakai
Slices	267	9312	2 %
Slice Flip-Flops	176	9312	1 %
4-Input LUTs	1.134	9312	12 %
Jalur IOBs	47	232	20 %
BUFGMUXs	4	24	16%

4.4.3 Mapping, Placing, dan Routing

Proses *mapping*,*placing* dan *routing* desain dalam arsitektur FPGA dilakukan oleh software Xilinx ISE Design Suite 11.1. Software melakukan pemetaan(*mapping*) jalur IOBs, menempatkan (*placing*) *gates* ke dalam CLBs, dan menghubungkan koneksi(*routing*) antar PSM.

4.4.4 Generate File Bitstream

Setelah proses *Mapping*, *Placing*, dan *Routing* selesai. Software akan menghasilkan *file bitstream* dengan tipe file *".bit"* yang akan di-*download* ke dalam arsitektur FPGA.

4.4.5 Download File Bitstream

Pada tahap ini *file bitstream* yang dihasilkan akan di-download ke hardware FPGA. Download dapat dilakukan dengan koneksi serial USB. Pada perancangan ini, proses download menggunakan kabel USB yang dihubungkan dengan perangkat bantu *Spartan-3 Starter Kit Board*.

BAB V
PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian tiap blok dan pengujian keseluruhan sistem mikroprosesor 8085 dan memori. Pengujian tiap blok dan pengujian keseluruhan sistem dilakukan untuk mengetahui apakah sistem yang telah dirancang bekerja sesuai dengan karakteristik yang diinginkan.

Pengujian yang dilakukan adalah pengujian terhadap:

1) Pengujian Clock
2) Pengujian Input-Output
3) Pengujian Memori
4) Pengujian Bidirectional
5) Pengujian Buffer
6) Pengujian Dekoder
7) Pengujian ALU dan Register Flag
8) Pengujian Unit Kontrol
9) Komparasi dengan Modul Praktikum Mikroprosesor 8085

5.1 Pengujian Clock

5.1.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah sistem pembagi clock telah berfungsi dengan tepat. Hal ini dilakukan untuk memastikan bahwa hasil clock sesuai dengan spesifikasi clock mikroprosesor 8085 yang dalam perancangan ini menggunakan nilai clock maksimumnya yaitu 3,25 MHz.

5.1.2 Peralatan Pengujian

Peralatan yang digunakan dalam pengujian unit *input output* ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) Logic Analyzer
4) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
5) *Software* Digilent Adept untuk antarmuka
6) Software ELAB-080 untuk mengetahui sinyal clock

5.1.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit *clock* ini adalah sebagai berikut:

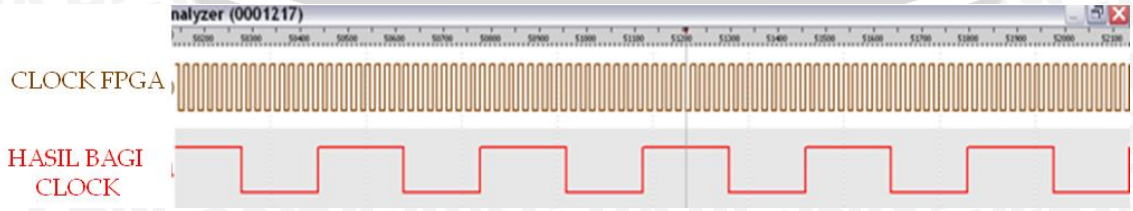
1) Menyiapkan peralatan yaitu FPGA, komputer, dan logic analyzer sehingga siap untuk melakukan pemprograman.
2) Menetapkan port untuk melihat logika keluaran baik logika *clock FPGA* maupun *clock* hasil pembagian clock.
3) Memuat program pembagi clock ke dalam FPGA.
4) Menyambungkan port yang ditentukan ke logic analyzer.
5) Menyambungkan *USB connection* logic analyzer ke komputer. Penyusunan perangkat ditunjukkan dalam Gambar 5.1.

Gambar 5. 1 Perangkat Pengujian Clock

6) Melihat hasil clock dalam software ELAB-080.

5.1.4 Data dan Analisis Hasil Pengujian Clock

Data hasil pengujian clock didapat dari hasil pembacaan logika clock oleh *logic analyzer*. Tampilan hasil keluaran clock ditunjukkan dalam Gambar 5.2.



Gambar 5. 2 Tampilan Clock FPGA dan Hasil Pembagian Clock pada Software ELAB-080

Berdasar data hasil pengujian clock didapatkan hasil bahwa clock hasil keluaran adalah 1/16 kali clock FPGA. Hal ini dapat dilihat bahwa saat clock FPGA telah mencapai 8 clock, clock keluaran baru akan berubah dari logika 0 menjadi 1 atau sebaliknya. Hal ini sesuai dengan rumus :

Clock keluaran = = = 3,25 MHz …….(1)

Hasil frekuensi clock keluaran adalah 3,25 MHz. Hal ini sesuai dengan clock yang akan digunakan dalam sistem mikroprosesor 8085 yang datanya diambil dari datasheet mikroprosesor 8085 bahwa frekuensi clock mikroprosesor 8085 adalah 3,25 MHz sehingga hasil clock ini dapat digunakan pada sistem.

5.2 Pengujian Input-Output

5.2.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah unit input dapat menerima masukan user saat kontrol IOR' diaktifkan yang selanjutnya data tersebut akan diproses dalam mikroprosesor dan apakah unit output dapat mengeluarkan keluaran yang benar saat kontrol IOW' diaktifkan.

5.2.2 Peralatan Pengujian

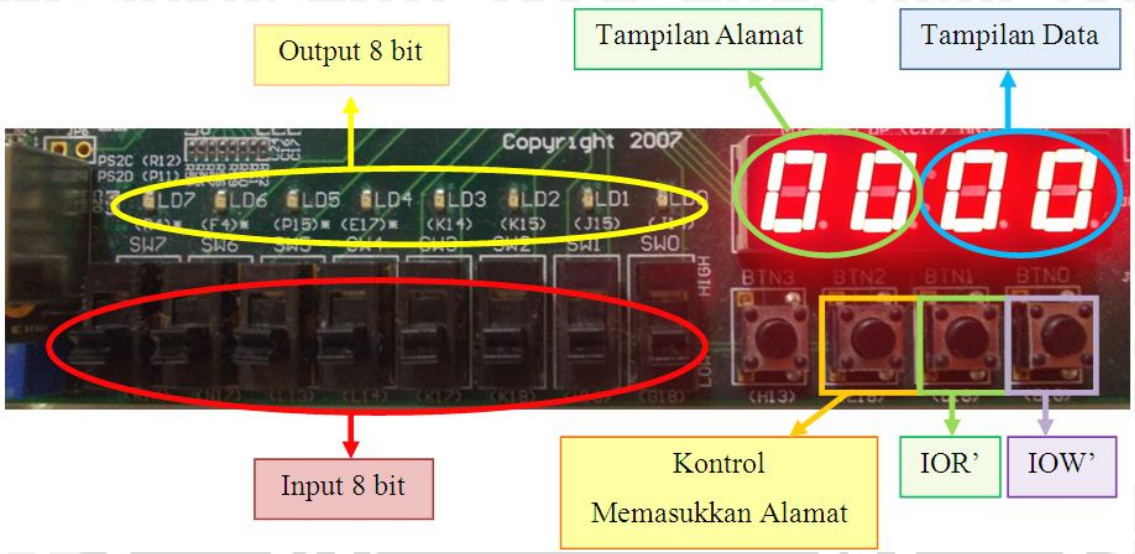
Peralatan yang digunakan dalam pengujian unit *input output* ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
4) *Software* Digilent Adept untuk antarmuka

5.2.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit *input output* ini adalah sebagai berikut:

1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman
2) Menetapkan *input output* dan sinyal kontrol IOR dan IOW yang akan digunakan pada FPGA sesuai dengan yang ditunjukkan dalam Gambar 5.3.



Gambar 5. 3 Penentuan Input, Output, dan Kontrol pada Pengujian Unit Input Output

3) Melakukan Pengujian dan Melihat Hasil Masukan maupun Keluaran Unit Input dan Output

Pada pengujian ini, input 8 bit berguna untuk dua fungsi. Pada awal penggunaan, input digunakan untuk meletakkan alamat. Saat *button* untuk memasukkan alamat ditekan, maka data yang ada pada input 8 bit akan tertampil pada tampilan alamat. Cara memasukkan alamat ditunjukkan dalam Gambar 5.4.

Gambar 5. 4 Menentukan Alamat dengan Menekan *Button* "*Active Address*"

Setelah itu, input 8 bit diatur sesuai dengan nilai data yang ingin kita masukkan. Sesuai dengan rancangan pada bab 4, input beralamatkan 12H sehingga data akan dapat tertampil di tampilan data jika angka pada tampilan alamat adalah 12 dan IOR' aktif.

Saat ingin mengeluarkan data yang telah tersimpan ke output 8 bit, user terlebih dahulu mengatur alamat agar bernilai 13H dengan cara yang sama saat mengatur alamat pada proses pemasukan data input. Setelah alamat pada tampilan data telah berganti

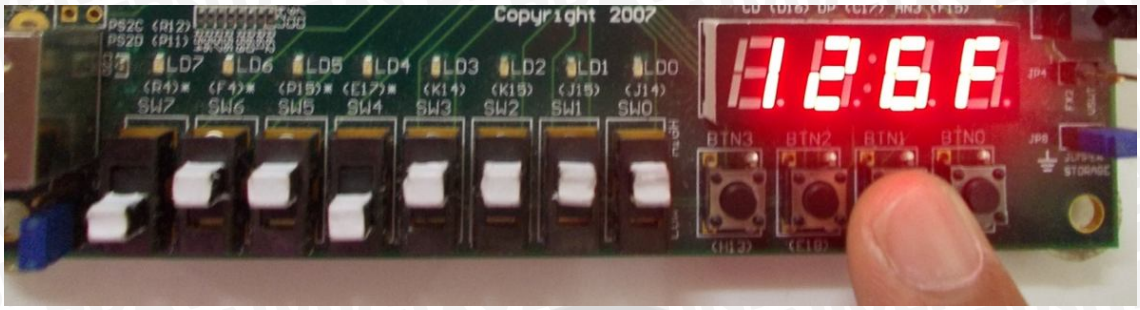
maka saat IOW' aktif data yang telah tersimpan akan keluar pada unit output 8 bit. Dalam pengujian ini, *seven segment* tampilan data dimisalkan sebagai data yang nantinya akan terdapat di dalam mikroprosesor.

5.2.4 Data Hasil Pengujian Unit Input Output

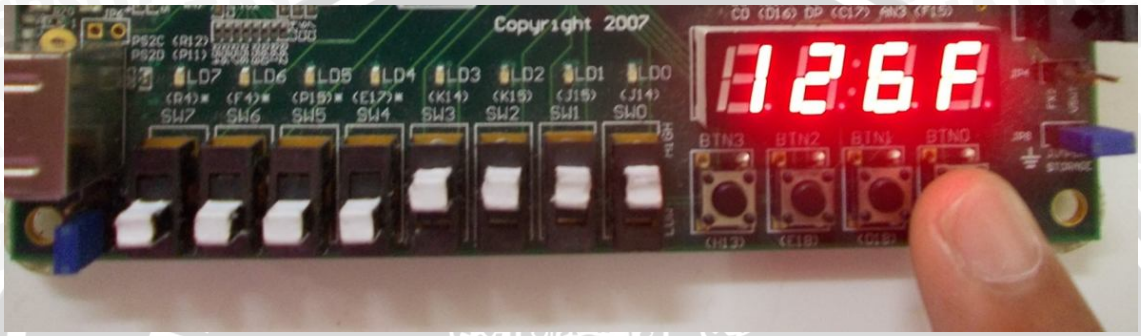
Pengujian dilakukan sebanyak 9 kali yaitu dengan menentukan alamat terlebih dahulu kemudian memberi masukan ke input 8 bit (switch) dan melihat hasilnya di output 8 bit sesuai dengan fungsi kontrol masing-masing. Data hasil pengujian ditunjukkan dalam Tabel 16 dan dokumentasi hasil pengujian ditunjukkan dalam Gambar 5.5.

Tabel 17 Data Hasil Pengujian Unit Input Output

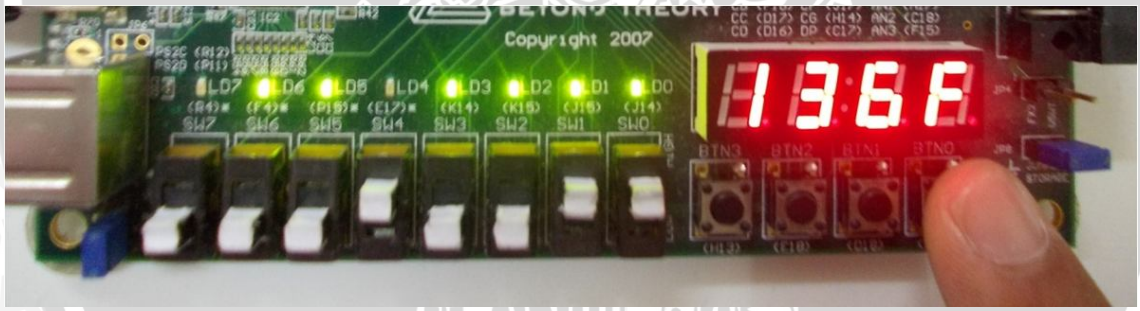
No.	Alamat (7 segment)	Input	IOR'	Data (7 segment)	IOW'	Output
1	00	00010101	aktif	00	non-aktif	off off off off off off off off
2	12	01101111	aktif	6F	non-aktif	off off off off off off off off
3	15	11010101	aktif	6F	non-aktif	off off off off off off off off
4	12	00001111	non-aktif	6F	aktif	off off off off off off off off
5	13	00010011	non-aktif	6F	aktif	off on on off on on on on
6	1A	01111110	aktif	6F	non-aktif	off on on off on on on on
7	12	01110010	aktif	72	non-aktif	off on on off on on on on
8	12	01110010	non-aktif	72	aktif	off on on off on on on on
9	13	00010011	non-aktif	72	aktif	off on on on off off on off


Copyright 2007
1.2.6.F.
BTN3 BTN2 BTN1 BTN0

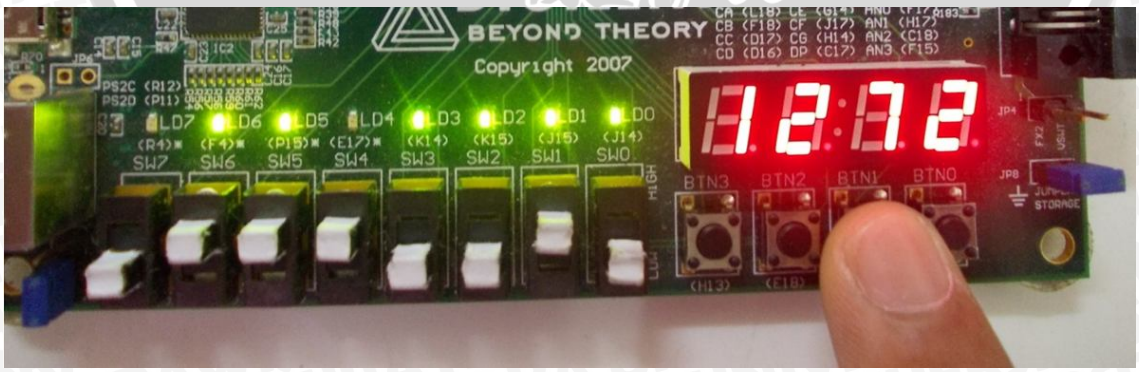
(a)


Copyright 2007
1.2.6.F.
BTN3 BTN2 BTN1 BTN0

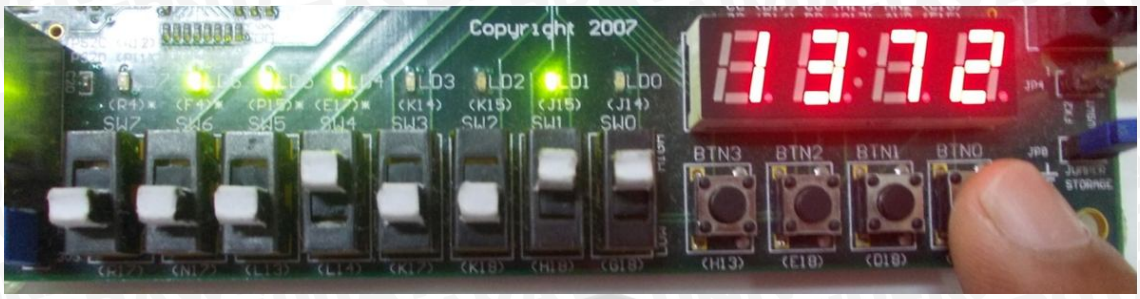
(b)


Copyright 2007
1.3.6.F.
BTN3 BTN2 BTN1 BTN0

(c)


BEYOND THEORY
Copyright 2007
1.2.7.2
BTN3 BTN2 BTN1 BTN0

(d)



(e)

Gambar 5. 5 (a) Hasil Pengujian Unit Input Output pada Tabel 16 no. 2

(b) no. 4 (c) no. 5 (d) no. 7 (e) no. 9

5.2.5 Analisis Hasil Pengujian Unit Input Output

Pada pengujian unit input output didapatkan hasil bahwa unit input telah dapat dibaca dan masuk menjadi data yang akan diolah mikroprosesor saat alamat diset 12H. Sedangkan output dapat ditulisi dengan data yang akan ditampilkan saat alamat diset 13H.

Kesesuaian rancangan unit input dengan hasil pengujian dibuktikan dengan sesuainya hasil tampilan 7 segment dengan input yang diberikan. Pada pengujian ke-2, kondisi *switch* bernilai 6FH. Saat 7-segment tampilan alamat bernilai 12H dan IOR' ditekan (aktif) maka angka yang dimasukkan dari switch input bernilai sama dengan tampilan data 7-segment. Sedangkan pada pengujian ke-3 nilai data 7-segment tidak berubah, tetap seperti nilai pada pengujian ke-2 karena alamat bernilai 15H. Alamat 15H tidak cocok dengan alamat yang telah diset sebagai alamat port input sehingga data switch input tidak akan masuk ke dalam sistem.

Kesesuaian rancangan unit output dengan hasil pengujian dibuktikan dengan sesuainya hasil tampilan 7-segment dengan hasil tampilan output. Pada pengujian ke-5, kondisi 7-segment data bernilai 6FH. Saat 7-segment tampilan alamat bernilai 13H dan IOW ditekan (aktif) maka angka yang tersimpan (tertampil dalam 7-segment data) akan ditampilkan pula pada tampilan LED output. Sedangkan pada pengujian ke-8 dibuktikan bahwa saat alamat tidak diset pada angka 13H, maka output akan tetap menampilkan data sebelumnya, tidak akan mengeluarkan data baru karena ketidakcocokan alamat port input dengan alamat yang dikirimkan oleh prosesor.

Dalam pengujian unit input output tidak terdapat kesalahan atau hasil *error*. Hal ini dibutuhkan agar mikroprosesor dapat membaca input dan menuliskan data ke output dengan tepat.

5.3 Pengujian Memori

5.3.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah unit memori dapat ditulisi dan dibaca sesuai dengan perintah yang diberikan dan juga sesuai dengan alamat yang diakses.

5.3.2 Peralatan Pengujian

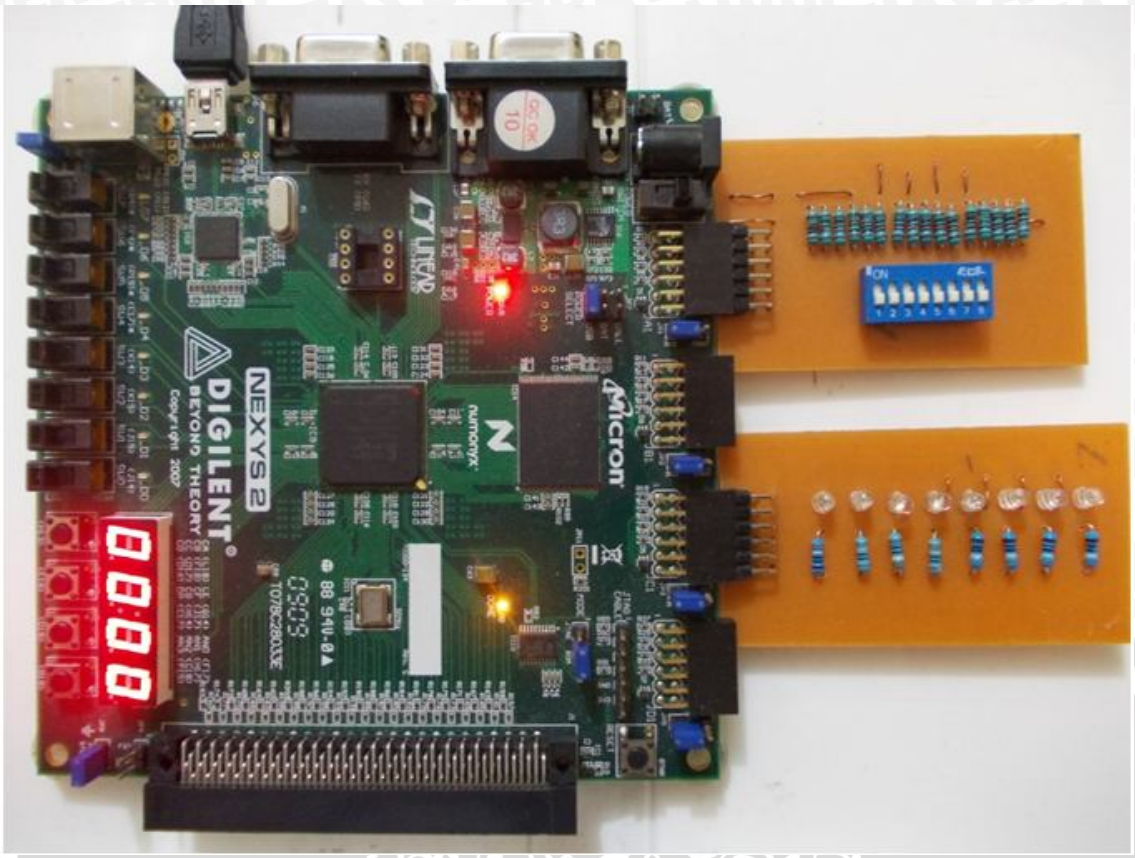
Peralatan yang digunakan dalam pengujian unit *input output* ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
4) *Software* Digilent Adept untuk antarmuka

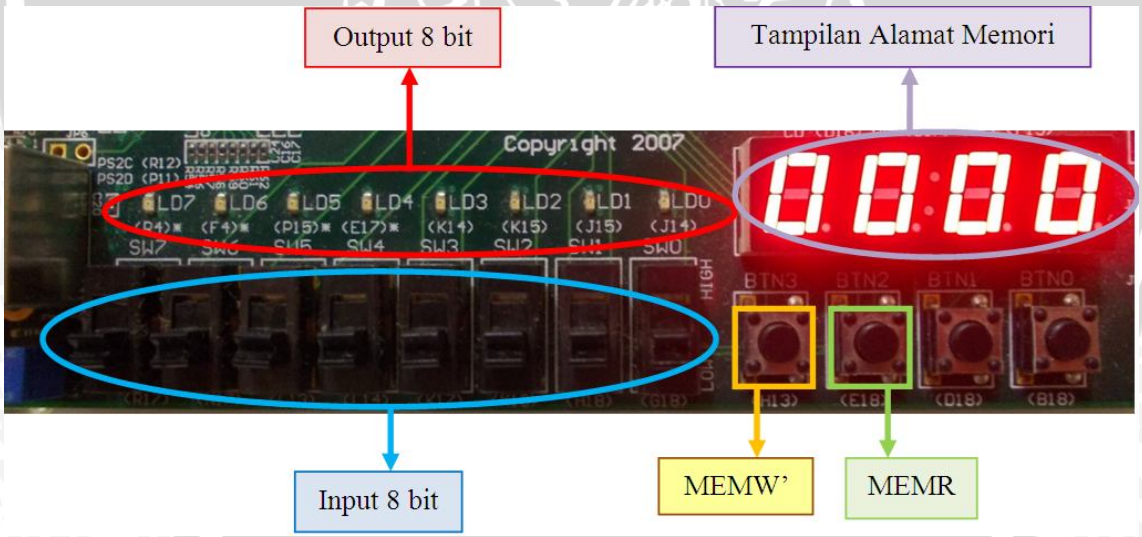
5.3.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit memori ini adalah sebagai berikut:

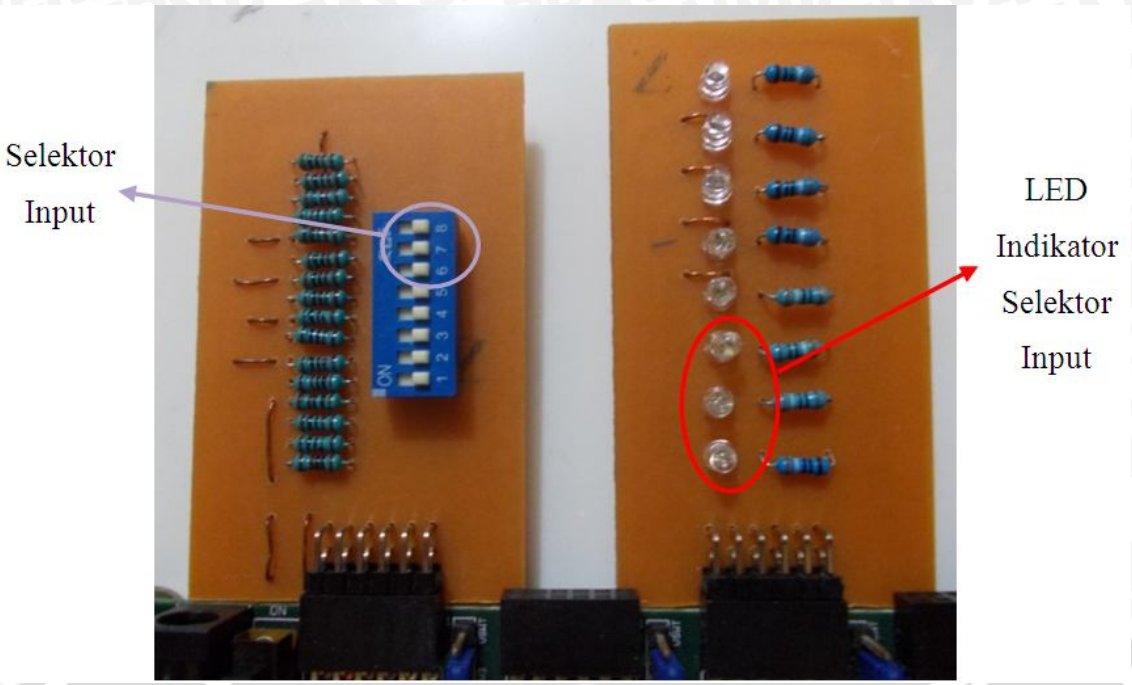
1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman
2) Menetapkan port input yang berfungsi memasukkan data alamat dan data yang akan ditulis ke memori dan juga menetapkan port output sebagai tampilan hasil keluaran serta sinyal kontrol MEMR’ dan MEMW’ yang akan digunakan pada FPGA. Dalam pengujian ini, switch dan LED indikator ditambahkan pada port tambahan yang berfungsi untuk menyeleksi kapan input digunakan untuk memasukkan alamat (0:7) , alamat (8:15), dan input data. Penetapan port-port ini ditunjukkan dalam Gambar 5.6 (a), (b), dan (c).


DIGILENT
NEXYS2
Copyright 2007
Micron

(a)


Output 8 bit
Tampilan Alamat Memori
Copyright 2007
LD7 LD6 LD5 LD4 LD3 LD2 LD1 LD0
SW7 SW6 SW5 SW4 SW3 SW2 SW1 SW0
BTN3 BTN2 BTN1 BTN0
Input 8 bit
MEMW'
MEMR

(b)



(c)

Gambar 5. 6 (a) Modul Pengujian Menyeluruh
(b) Penentuan Port Input, Output, Tampilan 7 Segment Alamat, dan Kontrol
(c) Tampilan Selektor Input dan LED Indikatornya

4) Melakukan Pengujian dan Melihat Hasil Masukan maupun Keluaran Unit Memori
5) Melihat Perubahan dan Hasil Pengujian

Pada pengujian ini, input 8 bit berguna untuk tiga fungsi tergantung selektor input. Pada awal penggunaan, input digunakan untuk meletakkan alamat (0:7) dilanjutkan dengan alamat (8:15). Setelah alamat tertampil di tampilan 7 segment alamat, input difungsikan sebagai data masukan. MEMR' adalah kontrol agar memori mengeluarkan data yang tersimpan di alamat yang tertampil di 7 segment sedangkan MEMW' adalah kontrol agar memori menyimpan data yang dimasukkan dari input 8 bit dan menyimpannya di alamat yang telah ditentukan atau tertampil di 7 segment.

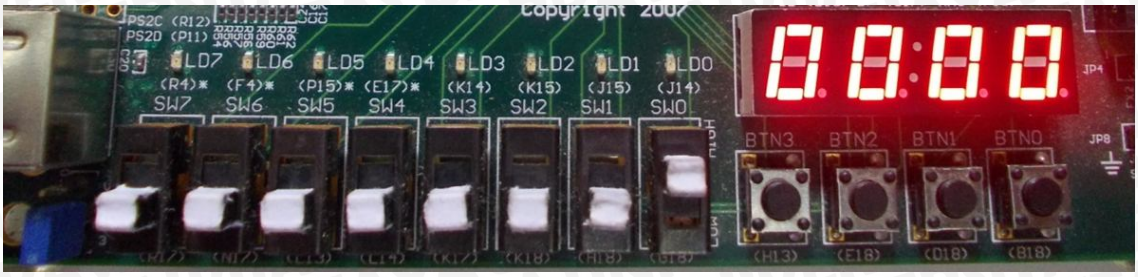
Pada pengujian ini, data input dan data output memori dipisahkan (tidak menjadi satu bagian) agar perubahan input dan output data memori lebih tampak dan dapat mengoreksi kesalahan dengan tepat.

5.3.4 Data Hasil Pengujian Unit Memori

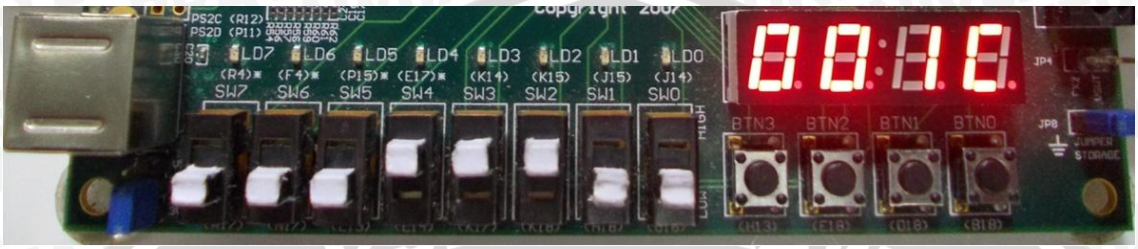
Pengujian dilakukan sebanyak 21 kali dengan cara yang telah dijabarkan sebelumnya. Data hasil pengujian ditunjukkan dalam Tabel 17. Data hasil pengujian diasumsikan bahwa data yang ada adalah data saat alamat telah tertampil dalam 7 segment dan selektor input diset “100” yang berarti input telah siap untuk menjadi data masukan. Dokumentasi hasil pengujian ditunjukkan dalam Gambar 5.7.

Tabel 18 Data Hasil Pengujian Unit Memori

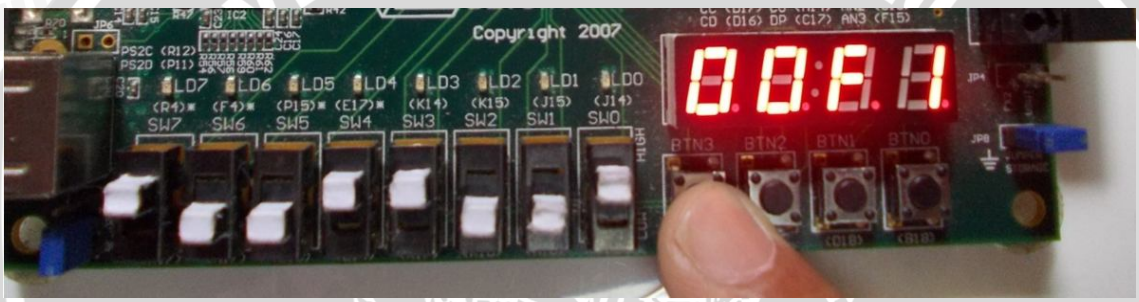
No.	Alamat (7 segment)	Input	MEMW’	MEMR’	Output
1.	0000	00000001	non-aktif	non-aktif	off off off off off off off off
2.	0001	00010001	aktif	non-aktif	off off off off off off off off
3.	0002	00100010	aktif	non-aktif	off off off off off off off off
4.	0011	00110101	aktif	non-aktif	off off off off off off off off
5.	001C	00011100	non-aktif	non-aktif	off off off off off off off off
6.	001C	00111100	aktif	non-aktif	off off off off off off off off
7.	0022	01000100	aktif	non-aktif	off off off off off off off off
8.	0033	00110100	aktif	non-aktif	off off off off off off off off
9.	00E2	01010010	non-aktif	aktif	off off off off off off off off
10.	00F1	10011001	aktif	non-aktif	off off off off off off off off
11.	00FF	10101010	aktif	non-aktif	off off off off off off off off
12.	0000	00110011	aktif	non-aktif	off off off off off off off off
13.	0001	11111111	non-aktif	aktif	off off off on off off off on
14.	0002	11111111	non-aktif	aktif	off off on off off off on off
15.	0011	11111111	non-aktif	aktif	off off on on off on off on
16.	00C0	11000000	non-aktif	aktif	off off off off off off off off
17.	001C	11111101	non-aktif	aktif	off off on on on on off off
18.	0000	11010101	non-aktif	aktif	off off on on off off on on
19.	00E2	11010101	non-aktif	aktif	off on off on off off on off
20.	00F1	00000000	non-aktif	aktif	on off off on on off off on
21.	00FF	11111111	non-aktif	aktif	on off on off on off on off



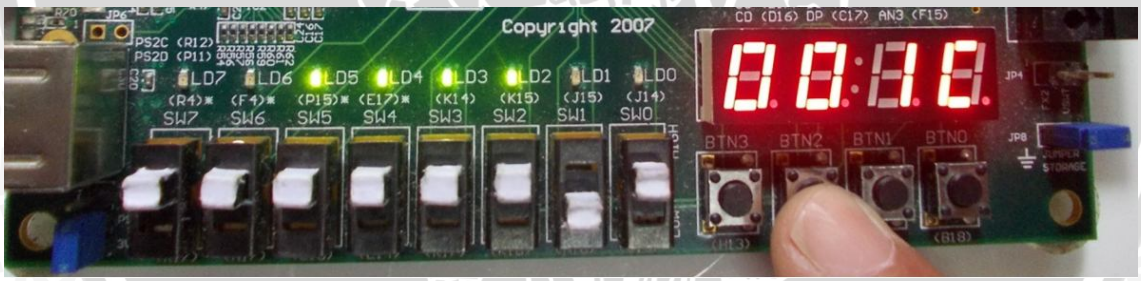
(a)



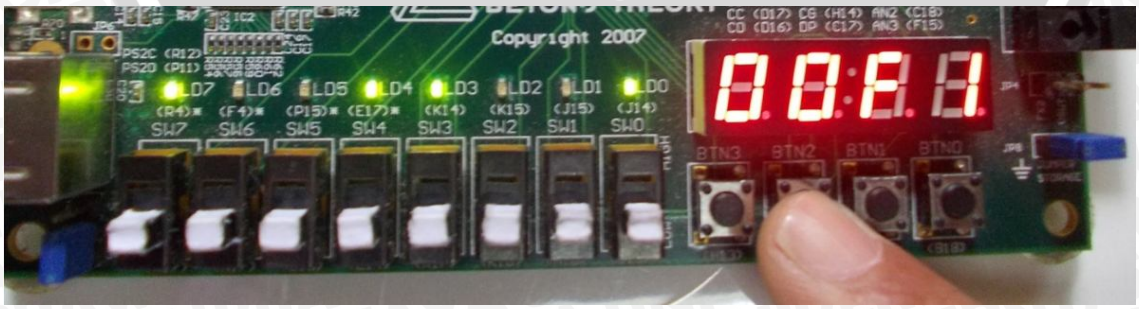
(b)



(c)



(d)



(e)

Gambar 5. 7 (a) Hasil Pengujian Unit Memori pada Tabel 17 no. 1

(b) no. 6 (c) no. 10 (d) no. 17 (e) no. 20

5.3.5 Analisis Hasil Pengujian Unit Memori

Pada pengujian unit memori didapatkan hasil bahwa memori telah dapat menerima masukan yang diberikan berupa data yang akan disimpan sesuai dengan alamat yang ditunjuk saat diberikan kontrol MEMW' dan memori juga dapat menampilkan data yang disimpan sesuai alamat yang ditunjuk dan saat diberikan kontrol MEMR'.

Hal ini dibuktikan dari hubungan pengujian ke-3 dan ke-9. Pada pengujian ke-3 dituliskan data (dengan menekan tombol MEMW') yang bernilai 3CH ke dalam memori yang beralamatkan 001C. Pengujian ke-9 adalah pengujian pembacaan memori beralamatkan 001C. Pada tampilan keluaran memori, hasil yang tertampil pada output LED adalah 3CH. Hal yang sama dapat dilihat pada hubungan antara pengujian ke-6 dan ke-0 serta pengujian ke-4 dan ke-11.

Sedangkan saat pengujian ke-8 adalah pengujian pembacaan memori dari memori yang beralamatkan 00C0. Saat proses pembacaan data yang tersimpan dalam memori tersebut, output LED menampilkan angka 00H. Hal ini disebabkan karena belum adanya data yang dituliskan dalam memori 00C0 tersebut. Pengujian pada alamat memori yang terakhir juga dilakukan yaitu pada memori yang beralamatkan 00FFH. Hal ini membuktikan bahwa kapasitas memori adalah 256 byte yaitu dapat menyimpan data pada rentang alamat 0000H-00FFH. Pada pengujian memori ini tidak terdapat kesalahan atau error pengujian.

5.4 Pengujian Unit Bidirectional

5.4.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah sistem unit bidirectional ini bekerja. Pada pengujian ini, pengujian dijadikan satu dengan pengujian latch karena sumber masukan dari dan ke mikroprosesor dari sumber yang sama. Pengujian ini dibutuhkan agar koneksi antara mikroprosesor dan memori ataupun dengan unit input dan output dapat terkomunikasi dengan baik.

5.4.2 Peralatan Pengujian

Peralatan yang digunakan dalam pengujian unit bidirectional ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320

2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA

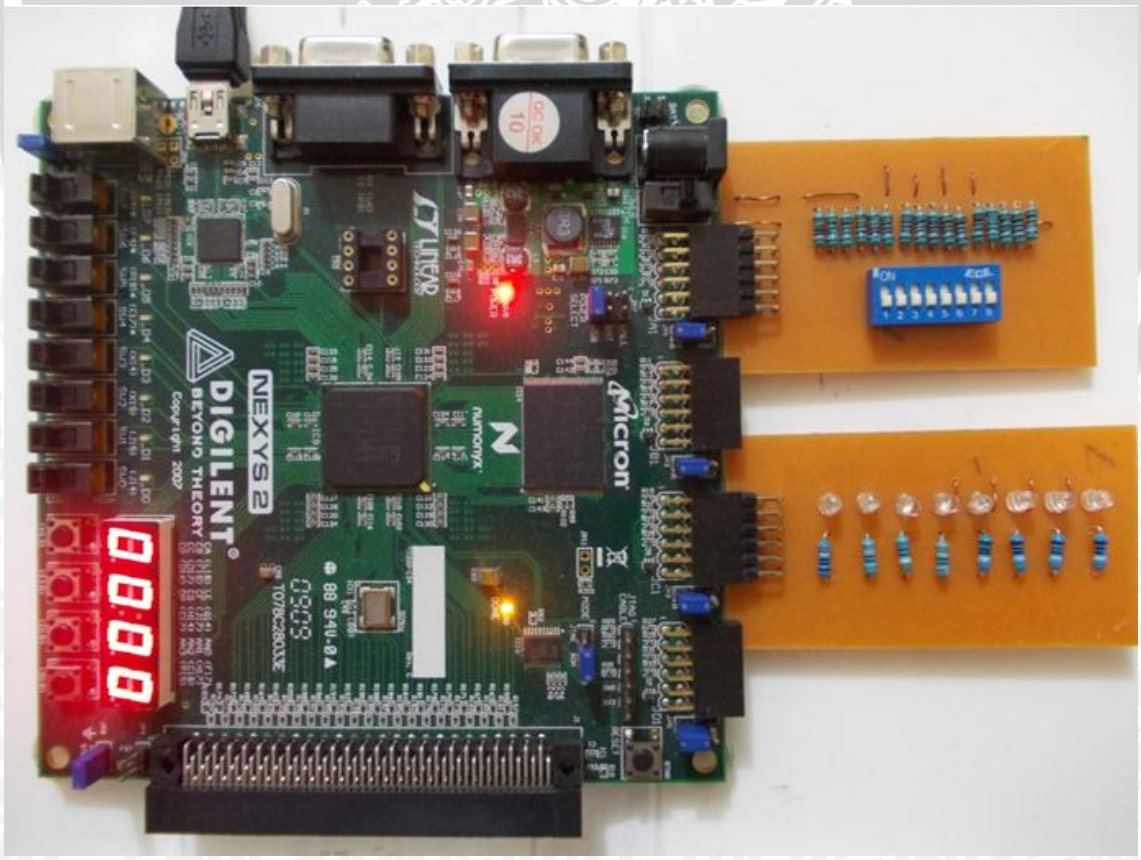
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman

4) *Software* Digilent Adept untuk antarmuka

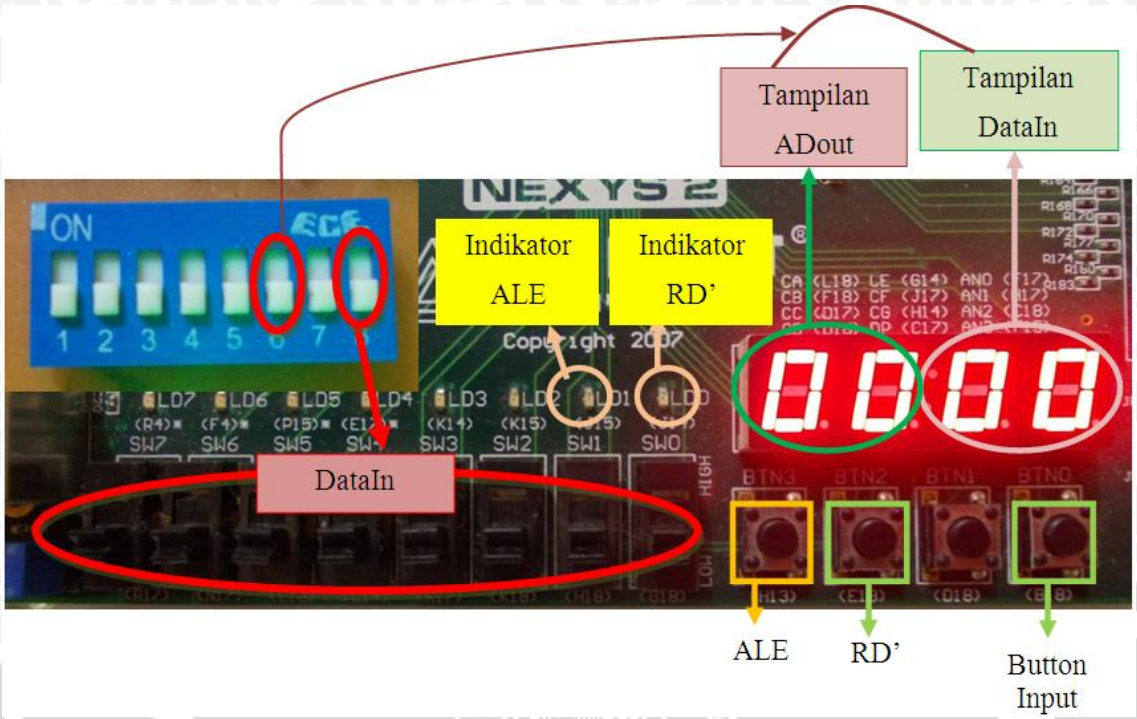
5.4.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit bidirectional ini adalah sebagai berikut:

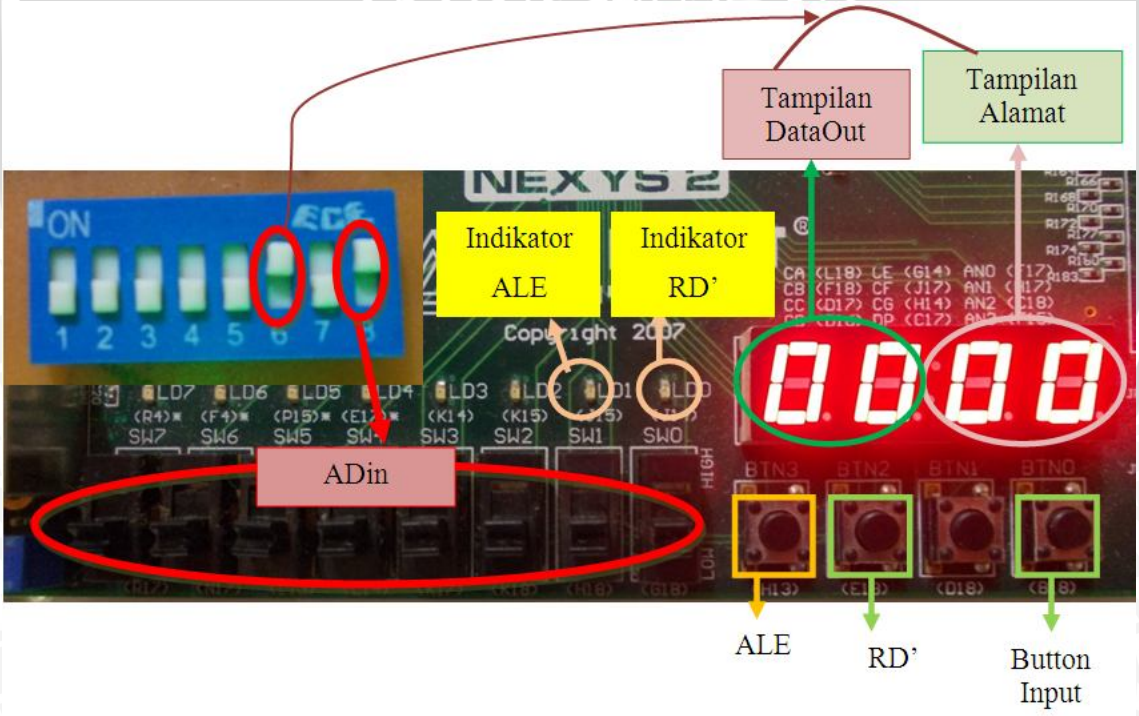
1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman
2) Menetapkan port input yang berfungsi masukan data untuk pin ADin (alamat data) dan DataIn. Button difungsikan untuk kontrol RD' dan ALE. menampilkan hasil operasi ALU dan kapan saatnya menampilkan hasil status flag. Dalam pengujian ini, switch ditambahkan pada port tambahan yang berfungsi sebagai pilihan untuk port input saat kapan akan mengisi AD dan saat kapan akan mengisi DataIn. Terdapat juga switch yang mengatur kapan saatnya display 7-segment akan menampilkan DataOut dan Alamat atau DataIn dan ADout. Penetapan port-port ini ditunjukkan dalam Gambar 5.8 (a), (b), dan (c).



(a)


Tampilan
ADout
Tampilan
DataIn
Indikator
ALE
Indikator
RD'
DataIn
ALE
RD'
Button
Input

(b)


Tampilan
DataOut
Tampilan
Alamat
Indikator
ALE
Indikator
RD'
ADin
ALE
RD'
Button
Input

(c)

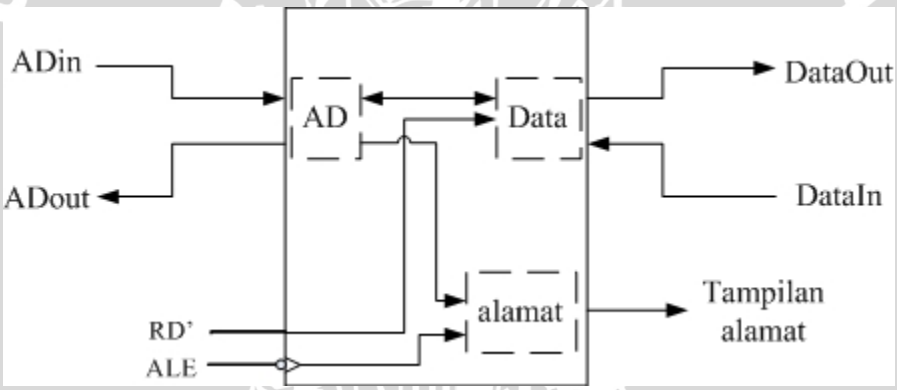
(d)

Gambar 5. 8 (a) Modul Pengujian Menyeluruh

(b) dan (c) Penentuan Input, Output dan Tampilan 7-segment berdasar Switch

(d) LED Indikator ADin (8 bit)

3) Membuat program sesuai dengan diagram blok pengujian unit bidirectional yang ditunjukkan dalam Gambar 5.9.



Gambar 5. 9 Diagram Blok Pengujian Bidirectional

4) Melihat Perubahan dan Hasil Saat Pengujian

Pada pengujian ini, pengujian bidirectional digabungkan dengan unit buffer untuk data alamat bit ke-8 hingga bit ke-15. Hal ini dilakukan untuk menghindari adanya bentrok data pada saat blok telah diintegrasikan pada sistem. AD atau yang disebut jalur *address*/data tergabung jadi satu baik digunakan untuk jalur mentransfer alamat ataupun mentransfer data sehingga pengujian dilakukan bersamaan.

Data masukan dan data keluaran baik pada jalur *address*/data maupun pada jalur data dipisahkan (tidak menjadi satu bagian) agar perubahan input dan output data dari maupun ke unit luar lebih tampak dan dapat mengoreksi kesalahan dengan tepat.

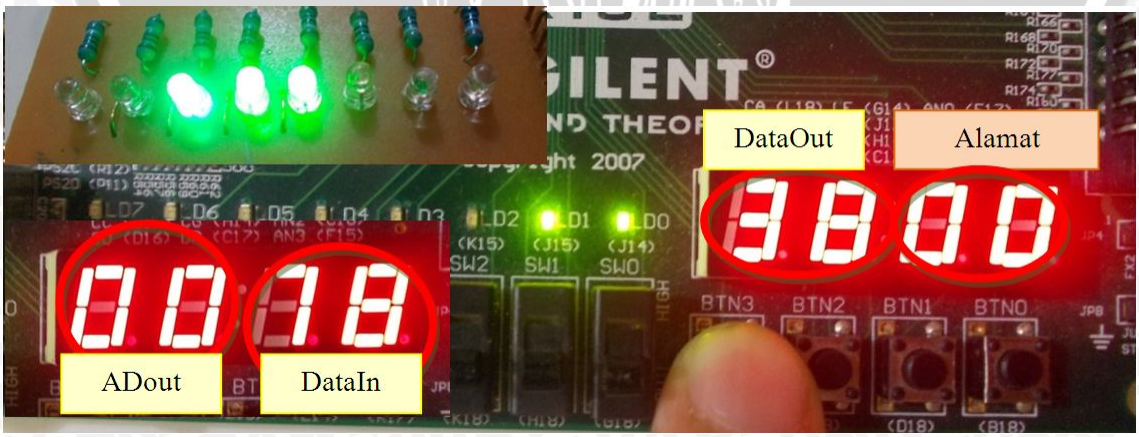
5.4.4 Data Hasil Pengujian Unit Bidirectional

Pengujian dilakukan sebanyak 8 kali yaitu dengan memasukkan data-data yang dibutuhkan dan melihat perubahannya pada 7-segment dan LED indikator. Data hasil pengujian ditunjukkan dalam Tabel 18 dan dokumentasi hasil pengujian ditunjukkan dalam Gambar 5.10.

Tabel 19 Data Hasil Pengujian Unit Bidirectional

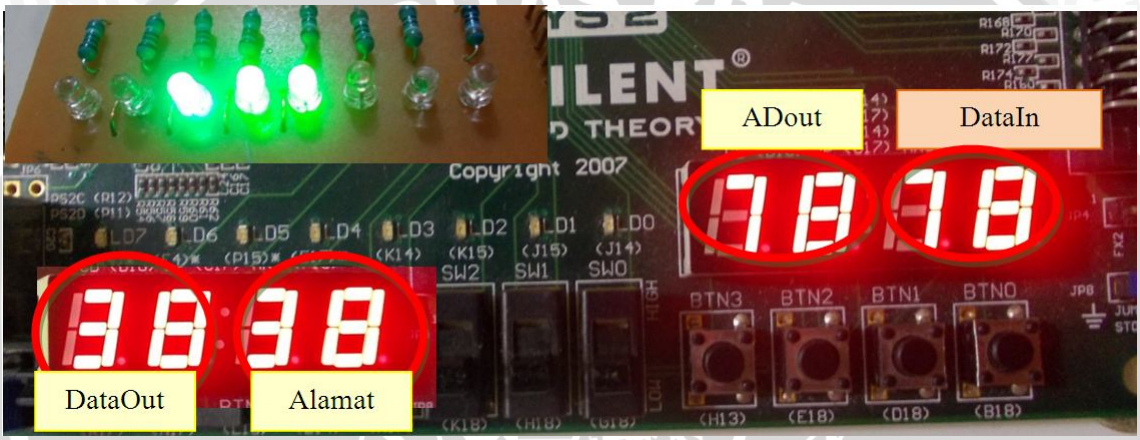
No.	ADIn	ALE	Alamat	RD'	DataOut	DataIn	ADout
1	00111000	0	00	1	38	78	00
2	00111000	1	00	1	38	78	00
3	00111000	0	38	1	38	78	00
4	00111000	0	38	0	38	78	78
5	01010101	0	38	1	55	01	78
6	01010101	1	38	1	55	01	78
7	01010101	0	55	1	55	01	78
8	00100101	0	55	0	55	01	01

(a)



(b)

(c)



(d)

Gambar 5. 10 (a) Hasil Pengujian Unit Bidirectional pada Tabel 18 no. 1
(b) no. 2 (c) no. 3 (d) no. 4

5.4.5 Analisis Hasil Pengujian Unit Bidirectional

Pada pengujian unit bidirectional didapatkan hasil bahwa unit ini dapat berfungsi sesuai dengan spesifikasi yang diinginkan. Saat ALE berlogika tepi turun maka data masukan (ADin) mengirimkan datanya ke port alamat. Pada pengujian ini, saat ALE berlogika tepi turun, sinyal RD' akan selalu dibuat berlogika 1. Hal itu sesuai dengan spesifikasi rancangan sistem mikroprosesor 8085.

Saat ALE tidak berlogika tepi turun, jika kontrol RD' aktif (dalam hal ini aktif adalah saat RD' berlogika 0) maka data yang terdapat pada DataIn akan dialirkan ke AD yang ditunjukkan dalam port ADout. Unit bidirectional ini dibutuhkan untuk menghubungkan AD, yaitu port yang terdapat pada mikroprosesor dengan unit luar seperti unit memori maupun unit input output.

Kesesuaian hasil pengujian dengan spesifikasi rancangan dibuktikan dari hasil pengujian. Pada pengujian ke-2 dan ke-3, terlihat adanya perpindahan kondisi ALE dari yang awalnya 1 menjadi 0. Pada saat pengujian ke-2, alamat masih 00, belum teraliri data, sedangkan saat pengujian ke-3 (kondisi ALE non aktif) alamat telah teraliri data dari AD. Hal ini telah membuktikan bahwa ALE yang telah dirancang untuk mengaktifkan aliran data ke alamat adalah berlogika tepi turun.

Saat kondisi kontrol RD' non aktif (berlogika 1) dan jika ALE tidak aktif maka secara otomatis data (untuk masukan terdapat dalam DataIn) akan mengalir ke AD yang ditunjukkan dalam tampilan port ADout sedangkan saat kondisi kontrol RD' aktif maka arahan data adalah dari Data ke AD (ditunjukkan dalam pengujian yaitu dari DataIn ke ADout).

Hal ini dapat dilihat pada perbandingan antara hasil pengujian ke-7 dan ke-8. Pada saat pengujian ke-7, RD' masih belum aktif sehingga arahan data adalah kearah DataOut atau dapat dikatakan keluar dari mikroprosesor (AD adalah koneksi port dari mikroprosesor) yang ditunjukkan dengan kesamaan data pada kolom ke-1 dan ke-5 yaitu sama-sama bernilai 01010101 (55H). Sehingga dapat diartikan bahwa aliran data adalah mengarah keluar mikroprosesor. Perbandingannya dapat dilihat dari data yang terdapat pada ADout. Data ini adalah data yang mengarah ke mikroprosesor. Data tersebut kondisinya masih tetap seperti sebelumnya, sehingga dapat disebut bahwa tidak ada aliran data yang menuju ke mikroprosesor.

Untuk hasil pengujian ke-8, RD' aktif, data mengarah ke dalam mikroprossor. Hal ini terlihat pada kesamaan antara DataIn dan ADout, sedangkan untuk DataOut tidak mengalami perubahan meskipun nilai ADin telah diubah. Pada pengujian unit bidirectional ini tidak terdapat kesalahan atau error pengujian.

5.5 Pengujian Unit Buffer

5.5.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah sistem unit bufffer ini dapat bekerja sesuai dengan perancangan yaitu saat ALE berlogika tepi turun, maka data input akan mengalir ke output yang nantinya disambungkan ke unit luaran mikroprosesor (unit input output maupun memori).

5.5.2 Peralatan Pengujian

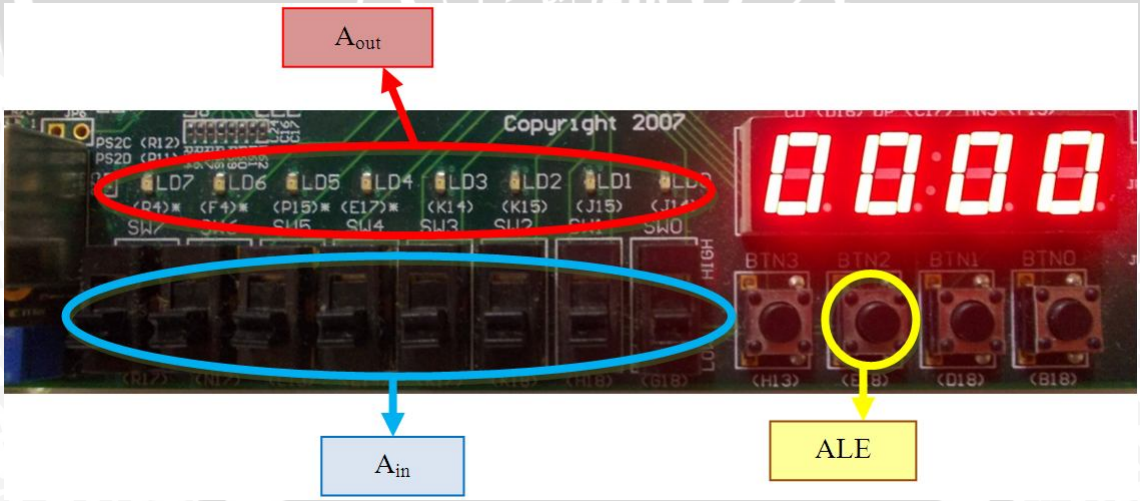
Peralatan yang digunakan dalam pengujian unit buffer ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
4) *Software* Digilent Adept untuk antarmuka

5.5.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit bidirectional ini adalah sebagai berikut:

1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman
2) Menetapkan port input yang berfungsi menjadi pin A (A adalah data keluaran mikroprosesor untuk mengirimkan alamat bit ke-8 hingga bit ke-15. Port output digunakan untuk melihat keluaran unit buffer. ALE digunakan untuk mengontrol kapan saatnya data input mengalir ke output. Penetapan port-port ini ditunjukkan dalam Gambar 5.11.



Gambar 5. 11 Penetapan Port Ain dan Aout

3) Melihat Perubahan dan Hasil Pengujian

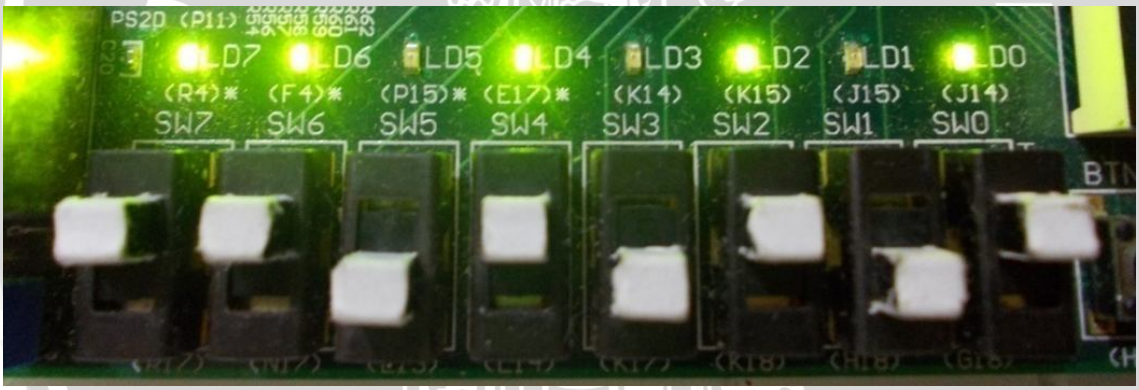
Pada pengujian ini yang dilakukan adalah melihat perubahan Aout berdasarkan logika yang diberikan ALE. Aout akan berubah saat ALE tepi turun dan jika tidak terdapat picu dari ALE maka nilai Aout akan tetap atau tidak akan berubah.

5.5.4 Data Hasil Pengujian Unit Buffer

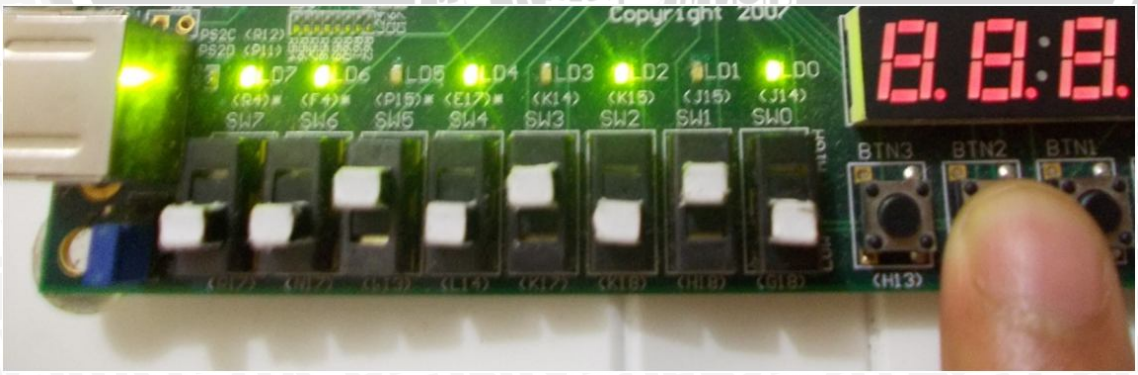
Pengujian dilakukan sebanyak 8 kali dengan melihat keluaran Aout saat Ain diubah-ubah dan juga berdasarkan kontrol ALE yang diberikan. Data Hasil Pengujian dan gambar hasil pengujian ditunjukkan dalam Tabel 19 dan Gambar 5.12.

Tabel 20 Data Hasil Pengujian Unit Buffer

No.	Ain	ALE	Aout
1	00101010	0	00000000
2	10101010	1	00000000
3	11010101	0	11010101
4	10101011	0	11010101
5	00101010	0	11010101
6	11010101	1	11010101
7	11001101	0	11001101
8	01100001	0	11001101



(a)



(b)

Gambar 5. 12 (a) Hasil Pengujian Unit Buffer pada Tabel 19 no. 3
(b) Hasil Pengujian Unit Buffer pada Tabel 19 no. 6

5.5.5 Analisis Hasil Pengujian Unit Buffer

Pada pengujian unit buffer didapatkan hasil bahwa unit ini dapat berfungsi sesuai dengan spesifikasi yang diinginkan. Saat ALE berlogika tepi turun maka data masukan (Ain) mengirimkan datanya ke port output (Aout). Spesifikasi rancangan ini dapat diterapkan pada sistem mikroprosesor 8085 untuk menyambungkan pin A (bit ke-8 hingga bit ke-15) ke unit luar mikroprosesor untuk mengalamati memori ataupun unit input output. Hal itu sesuai dengan spesifikasi rancangan sistem mikroprosesor 8085.

Kesesuaian ini dibuktikan melalui data hasil pengujian pada Tabel 5. Pada pengujian ke-3 terdapat perubahan sinyal kontrol ALE dari 1 ke 0 sehingga hasil Aout akan sama dengan Ain, begitu juga pada pengujian ke-7. Selain logika tepi turun ALE, maka Aout akan sama dengan hasil sebelumnya. Pada pengujian ini tidak terdapat error atau kesalahan pengujian.

5.6 Pengujian Unit Dekoder

5.6.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah sistem unit dekoder dapat memproses sinyal-sinyal masukan berupa IO/M' , RD' dan WR' menjadi sinyal yang dibutuhkan yaitu sinyal MEMR', MEMW', IOR', dan IOW'. Sinyal-sinyal inilah yang digunakan untuk mengakses unit luar mikroprosesor.

5.6.2 Peralatan Pengujian

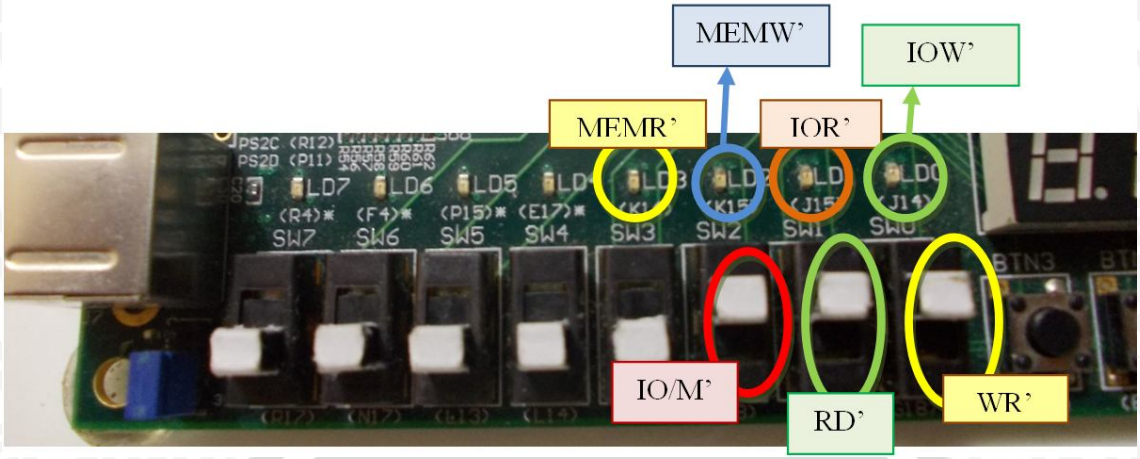
Peralatan yang digunakan dalam pengujian unit buffer ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
4) *Software* Digilent Adept untuk antarmuka

5.6.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit dekoder ini adalah sebagai berikut:

1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman
2) Menetapkan port input yang berfungsi menjadi pin IO/M', RD', dan WR'. Port output digunakan untuk melihat hasil keluaran sinyal MEMR', MEMW, IOR', dan IOW'. Penetapan port-port ini ditunjukkan dalam Gambar 5.13.



Gambar 5. 13 Penentuan Port-port untuk Pengujian Unit Dekoder

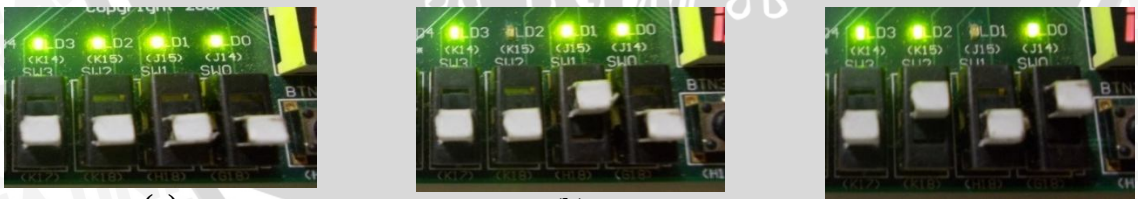
3) Melihat Perubahan dan Hasil Pengujian

5.6.4 Data Hasil Pengujian Unit Dekoder

Pengujian dilakukan sebanyak 8 kali dengan melihat keluaran sinyal-sinyal saat IO/M', RD', dan WR' diubah-ubah. Data hasil pengujian unit dekoder ditunjukkan dalam Tabel 20.

Tabel 21 Data Hasil Pengujian Unit Dekoder

No.	IO/M'	RD'	WR'	MEMR'	MEMW'	IOR'	IOW'
1	0	0	0	1	1	1	1
2	0	0	1	0	1	1	1
3	0	1	0	1	0	1	1
4	0	1	1	1	1	1	1
5	1	0	0	1	1	1	1
6	1	0	1	1	1	0	1
7	1	1	0	1	1	1	0
8	1	1	1	1	1	1	1



(a) (b) (c)

Gambar 5. 14 (a) Hasil Pengujian Unit Input Output pada Tabel 20 no. 1, (b) no. 3, (c) no. 6

5.6.5 Analisis Hasil Pengujian Unit Dekoder

Pada pengujian unit dekoder didapatkan hasil bahwa unit ini dapat berfungsi sesuai dengan spesifikasi yang diinginkan. Seluruh port pada unit ini akan aktif saat berlogika rendah. Pengujian dilakukan berdasarkan kemungkinan-kemungkinan yang ada saat unit dekoder berdiri sendiri atau tidak terhubung dengan unit lain.

Pada pengujian tersebut dibuktikan bahwa saat IO/M' berlogika 0 dan salah satu diantara WR' dan RD' aktif, maka dekoder akan mengirimkan sinyal kepada memori, baik MEMR' maupun MEMW' sedangkan saat IO/M' berlogika 1 dan salah satu diantara WR' dan RD' aktif, maka dekoder akan mengirimkan sinyal kepada unit input atau output, baik IOR' maupun IOW'.

Saat terhubung dengan mikroprosesor, mikroprosesor tidak akan pernah mengirimkan sinyal RD' dan WR' aktif secara bersamaan sehingga pada pengujian dekoder ini, saat RD' dan WR' aktif secara bersamaan, maka sinyal keluaran tidak ada satupun yang aktif. Pada pengujian unit buffer ini tidak terdapat kesalahan atau error pengujian.

5.7 Pengujian ALU

5.7.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah sistem ALU dapat bekerja sesuai dengan spesifikasi dan fungsi yang ditetapkan. ALU yang dirancang harus dapat melakukan operasi aritmatika dan logika secara dasar sehingga dapat menunjang kebutuhan tiap instruksi yang membutuhkan ALU untuk proses operasinya.

5.7.2 Peralatan Pengujian

Peralatan yang digunakan dalam pengujian unit ALU ini antara lain adalah:

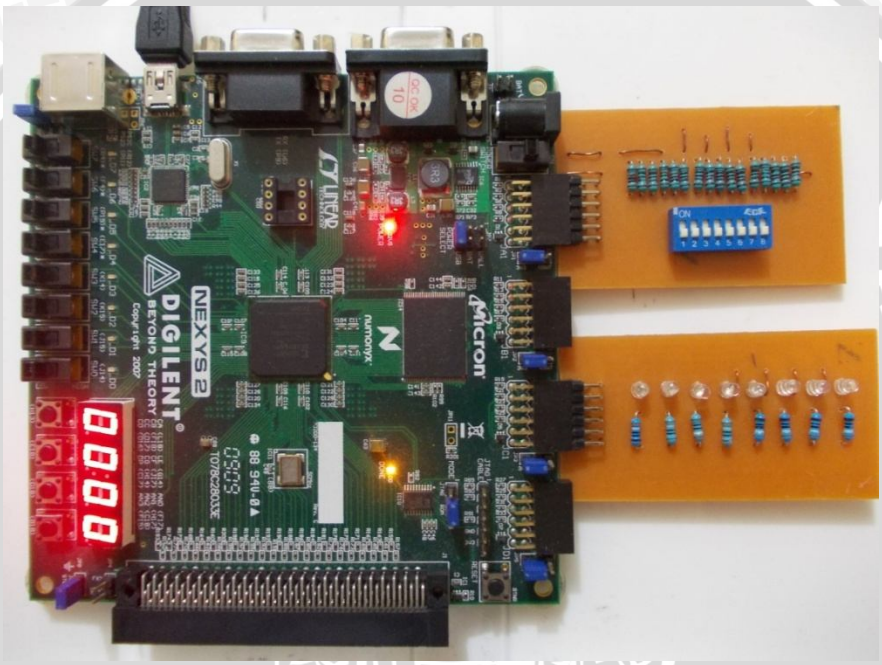
1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
4) *Software* Digilent Adept untuk antarmuka

5.7.3 Prosedur Pengujian

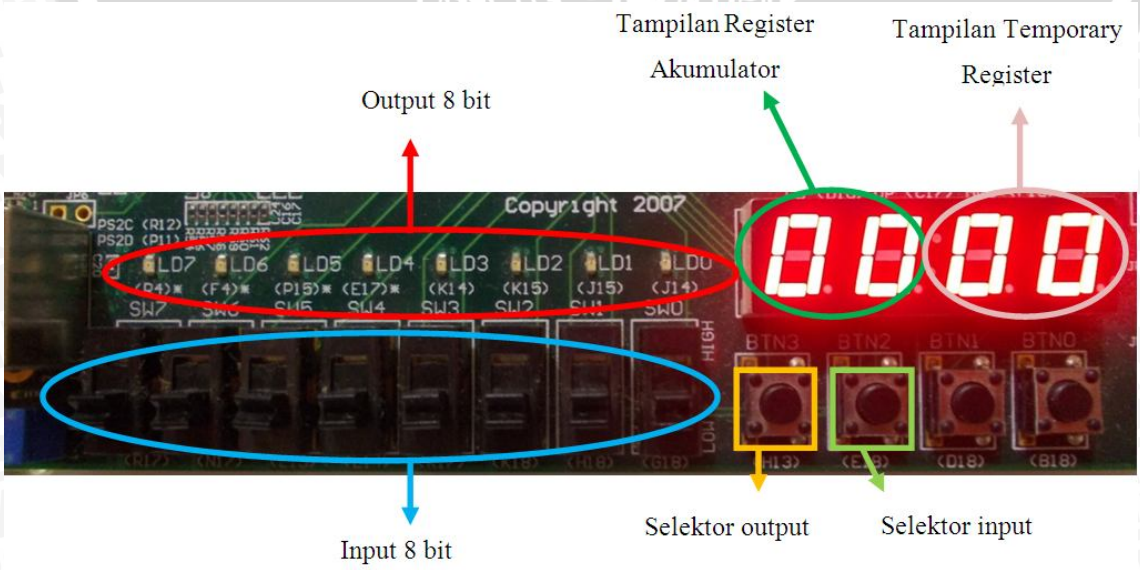
Prosedur dalam melakukan pengujian unit ALU ini adalah sebagai berikut:

1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman

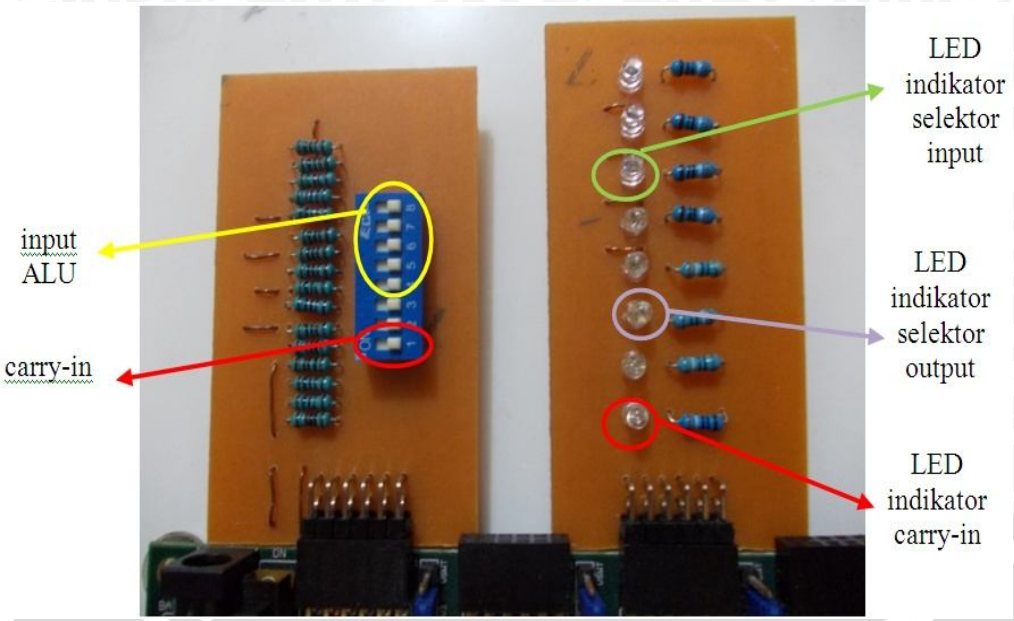
2) Menetapkan port input yang berfungsi memasukkan data register A dan register sementara (temp_reg) dan kedua data tersebut akan tertampil di 7-segment. Button difungsikan untuk memilih kapan saatnya output 8 bit menampilkan hasil operasi ALU dan kapan saatnya menampilkan hasil status flag. Dalam pengujian ini, switch ditambahkan pada port tambahan yang berfungsi sebagai input ALU yaitu input yang memberikan perintah kepada ALU untuk melakukan suatu operasi. LED indikator ditambahkan pada port tambahan yang berfungsi untuk menandai kapan saatnya output menjadi penampil hasil ALU atau status. Penetapan port-port ini ditunjukkan dalam Gambar 5.15 (a), (b), dan (c).



(a)



(b)



(c)

Gambar 5.15 (a) Modul Pengujian Menyeluruh

(b) Penentuan Input, Output, Tampilan Register Akumulator, Temporary Register, Selektor Input dan Output

(c) Tampilan switch input ALU dan carry-in serta LED Indikator Selektor Input, Output dan Carry-in

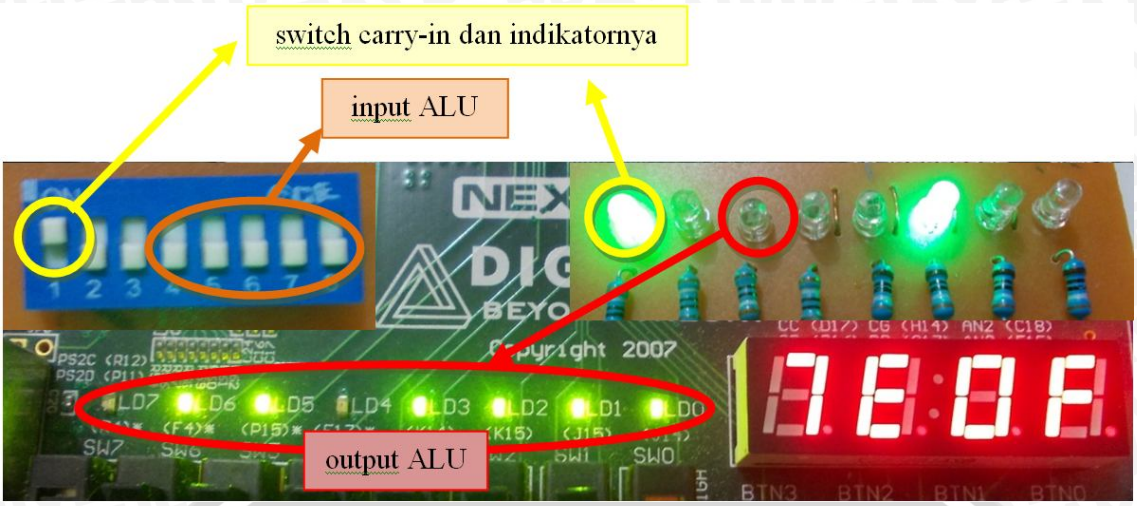
3) Melakukan Pengujian dan Melihat Hasil Keluaran Unit ALU
4) Melihat Perubahan dan Hasil Pengujian

5.7.4 Data Hasil Pengujian Unit ALU

Pengujian dilakukan sebanyak 37 kali yaitu dengan memasukkan data register akumulator dan temporary register kemudian hasilnya ditampilkan yaitu hasil output ALU dan register flag yang ditampilkan secara bergantian bergantung aktifnya selektor output. Hasil bergantung pula pada input ALU yaitu input yang mengirimkan perintah operasi untuk ALU. Data hasil pengujian ditunjukkan dalam Tabel 21 dan dokumentasi hasil pengujian ditunjukkan dalam Gambar 5.16.

Tabel 22 Data Hasil Pengujian Unit ALU

No.	input ALU	carry-in	Register Akumulator	Temporary Register	Output ALU	Register Flag
1	00000	0	7E	0F	01110000	00010000
2	00000	1	7E	0F	01110000	00010100
3	00000	0	0F	7E	10010001	10000001
4	00001	1	F1	7A	11110010	10000000
5	00001	0	BE	BE	10111111	10000000
6	00010	0	3C	A6	00111011	00000000
7	00010	1	98	68	10010111	10000000
8	00011	0	BE	82	00111100	00000100
9	00011	1	BE	82	00111011	00000000
10	00100	0	A6	BA	10111011	10000100
11	00100	1	CA	45	01000011	00000000
12	00101	0	FE	4D	01001100	00000000
13	00101	0	A6	BA	10111001	10000000
14	00110	0	A6	FF	10100101	10010101
15	00110	1	A6	FF	10100101	10010101
16	00111	0	83	CF	01010010	00010001
17	00111	1	83	CF	01010011	00010101
18	10000	0	4F	CF	10110000	10000000
19	10000	0	EA	EE	00010101	00010000
20	10001	0	B4	B0	01001111	00000000
21	10001	0	B4	88	01110111	00000100
22	10010	0	B4	88	10000000	10000000
23	10010	0	67	C2	01000010	01000010
24	10011	0	20	07	00100111	00010100
25	10011	0	19	89	10011001	10000100
26	10100	0	06	03	00000101	00000100
27	10100	0	1A	0E	00010100	00000100
28	10101	1	1A	0E	00110100	00000000
29	10101	1	23	06	01000110	00000000
30	10110	1	3F	21	10011111	10010100
31	10110	0	37	60	10011011	10000000
32	11000	1	8C	24	00011001	00000001
33	11000	0	8C	24	00011000	00000101
34	11001	1	8C	24	11000110	10000100
35	11001	0	8C	24	01000110	00000000


switch carry-in dan indikatornya
input ALU
output ALU

(a)


switch carry-in dan indikatornya
input ALU
output ALU

(b)


switch carry-in dan indikatornya
input ALU
Register Flag

(c)

(d)

(e)

Gambar 5. 16 Hasil Pengujian Unit ALU pada Tabel 2

(a) no.2 Saat Output Menunjukkan keluaran ALU

(b) no. 4 Saat Output Menunjukkan keluaran ALU

(c) no. 4 Saat Output Menunjukkan Register Flag

(d) no. 24 Saat Output Menunjukkan keluaran ALU

(e) no. 24 Saat Output Menunjukkan Register Flag

5.7.5 Analisis Hasil Pengujian Output ALU

Pada pengujian ke-1, ke-2, dan ke-3 input ALU berlogika 00000 yang berarti ALU melakukan operasi aritmatika pengurangan register A dengan register temporary. Pada pengujian ke-1 dan ke-2, register A bernilai 7EH yang dalam bilangan biner berarti 01111110 dan register temporary bernilai 0FH (00001111). Output ALU menunjukkan hasil 01110000 yang menunjukkan hasil pengurangan register A dan

register temporary dan hasil tersebut sesuai dengan penghitungan manual pengurangan register A dan register temporary. Proses perhitungan pengujian ke-1, ke-2, dan ke-3 dijabarkan dalam proses perhitungan berikut.

RegA	=>	01111110		RegA	=>	00001111	
TempReg	=>	00001111	-	TempReg	=>	01111110	-
Output ALU (uji ke-1 dan ke-2)	=>	01110000		Output ALU (uji ke-3)	=>	10010001	

Pada perhitungan tersebut diperlihatkan bahwa carry-in tidak digunakan dalam proses perhitungan ini karena input ALU 0000 berfungsi untuk mengurangkan register A dengan register temporary tanpa menggunakan carry-in sehingga brapapun nilai carry-in tidak akan berpengaruh terhadap keluaran operasi ALU.

Pada pengujian ke-4 dan ke-5, input ALU berlogika 00001 yang berarti ALU melakukan operasi aritmatika increment register A sehingga output ALU adalah hasil register A ditambahkan dengan 1.

RegA	=>	11110001		RegA	=>	10111110	
		1	+			1	+
Output ALU (uji ke-4)	=>	11110010		Output ALU (uji ke-5)	=>	10111111	

Perilaku yang sama juga terjadi pada pengujian ke-10 dan ke-11 namun yang dikurangi bukanlah register A namun register temporary.

TempReg	=>	10111010		TempReg	=>	01000101	
		1	+			1	+
Output ALU (uji ke-10)	=>	10111011		Output ALU (uji ke-11)	=>	01000110	

Pada pengujian ke-6 dan ke-7, input ALU berlogika 00010 yang berarti ALU melakukan operasi *decrement* register A yang artinya register A dikurangi dengan 1. Pada pengujian ke-6 nilai register A 3C (00111100) sehingga jika dikurangi 1 hasilnya adalaah 00111011 yang sesuai dengan hasil keluaran ALU yang tampak dalam kolom output ALU. Perilaku *decrement* terdapat pula saat pengujian ke-12 dan ke-13 yaitu saat ALU berlogika 00101 yang berarti ALU melakukan operasi decrement pada isi dalam register temporary.

RegA	=>	00111100		RegA	=>	10011000	
		1	-			1	-
Output ALU (uji ke-6)	=>	00111011		Output ALU (uji ke-7)	=>	10010111	

TempReg	=>	01001101		TempReg	=>	10111010	
		1	-			1	-
Output ALU (uji ke-12)	=>	01001100		Output ALU (uji ke-13)	=>	10111001	

Perilaku yang sama juga terjadi pada pengujian ke-12 dan ke-13 namun yang dikurangi bukanlah register A namun register temporary.

Pada pengujian ke-8 dan ke-9, input ALU berlogika 00011 yang artinya ALU melakukan operasi pengurangan register A dengan register temporary dan carry-in. Pada operasi ini, carry-in mempengaruhi hasil keluaran ALU karena nilai carry-in diperhitungkan. Pada pengujian ke-9, carry-in bernilai 0, isi register A adalah 10111110 dan isi register temporary adalah 10000010 dan carry-in bernilai 1. Proses perhitungan pada pengujian ke-9 dijelaskan sebagai berikut.

RegA	=>	10111110		Hasil1	=>	00111100	
TempReg	=>	10000010	-	carry-in	=>	1	-
Hasil1	=>	00111100		out_alu	=>	00111011	

Hasil keluaran ALU menunjukkan hasil yang sama dengan sistem perhitungan di atas. Dalam pengujian ALU ini, carry out maupun borrow out hasil perhitungan operasi aritmatika tidak ditampilkan dan akan dibahas secara terpisah dalam blok register flag.

Pengujian ke-14 dan ke-15 input ALU berlogika 0110 yang berarti melakukan proses penjumlahan register A dan register temporary dalam ALU. Pada pengujian ke-14, register A bernilai A6H (10100110) dan register temporary bernilai FFH (11111111). Output ALU menunjukkan hasil 10100101 yang sesuai dengan hasil sistem penjumlahan dua bilangan biner 8 bit. Untuk pengujian ke-16 dan ke-17 yaitu saat input ALU berlogika 00111, ALU melakukan proses penjumlahan dengan memperhitungkan carry sehingga output ALU adalah hasil penjumlahan register A, register temporary, dan carry-in. Prose perhitungan pada pengujian ke-17 dijelaskan sebagai berikut.

RegA	=>	10000011		Hasil1	=>	01010010	
TempReg	=>	11001111	+	carry-in	=>	1	+
Hasil1	=>	01010010		out_alu	=>	01010011	

Hasil keluaran ALU menunjukkan hasil yang sama dengan perhitungan diatas sehingga dapat dikatakan bahwa proses ALU saat input ALU berlogika 00111 berjalan dengan benar.

Pengujian ke-18 hingga ke-37 adalah pengujian operasi ALU untuk operasi logika. Pada pengujian ke-18 dan ke-19 yaitu saat input ALU berlogika 10000, output rALU adalah complement dari isi register A. Pada pengujian ke-18, register A berisi

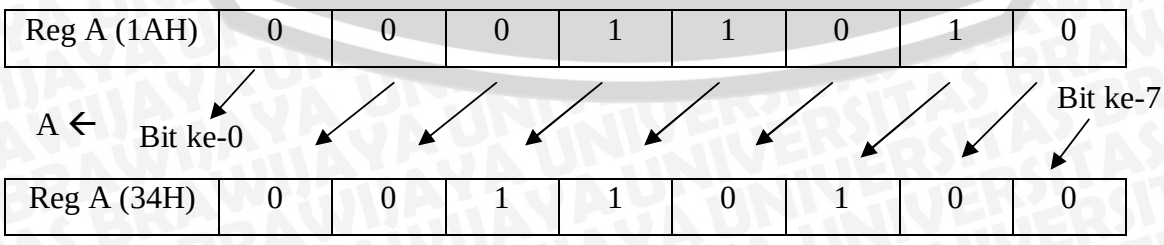
01001111 sehingga output ALU bernilai 10110000. Pengujian ke-20 adalah proses yang sama dengan pengujian ke-21 namun yang dicomplementkan bukanlah register A melainkan register temporary. Isi register temporary adalah B0H (10110000) dan output ALU bernilai 01001111 yang menunjukkan hasil complement register temporary.

Pada pengujian ke-22 dan ke-23 saat input ALU berlogika 10010 adalah saat ALU memproses register A yang di-AND-kan dengan register temporary. Pada pengujian ke-22, register A berisi B4H (10110100) dan register temporary berisi 88H (10001000). Jika register A di-AND-kan dengan register temporary hasilnya akan bernilai 10000000. Hasil tersebut sama dengan hasil output ALU yang menunjukkan keluaran fungsi 'RegA AND Temporary Register'.

Pengujian ke-24 dan ke-25 adalah pengujian saat input ALU berlogika 10011 yang berarti proses register A di-OR-kan dengan register temporary. Register A berisi 20H (001010111) dan register temporary berisi 07H (00000111). Jika register A di-OR-kan dengan register temporary hasilnya akan bernilai 00100111. Hasil tersebut sama dengan hasil output ALU yang menunjukkan keluaran fungsi 'RegA ORTemporary Register'.

Pengujian ke-26 dan ke-27 adalah pengujian saat input ALU berlogika 10100 yang berarti ALU melakukan operasi XOR, yaitu register A di-XOR-kan dengan register temporary. Register A berisi 06H (00000110) dan register temporary diisi dengan nilai 03H (00000011). Karakteristik XOR adalah jika bilangan yang di-XOR-kan sama, maka hasilnya adalah 0, sedangkan jika kebalikannya maka hasilnya adalah 1. Bilangan dalam pengujian sama dengan pengoperasian manual dan hasil XOR tersebut adalah 00000101.

Pengujian ke-28 sampai ke-37 ditujukan untuk menguji ALU saat memproses perintah bitwise. Untuk pengujian ke-28 dan ke-29 digunakan untuk menggeset 1 bit nilai register akumulator ke kiri. Pada pengujian ke-28, nilai akumulator 1AH menjadi 34H.



Hal ini membuktikan bahwa perintah untuk menggeser akumulator ke kiri dapat berjalan.

Pengujian ke-30 dan ke-31 menunjukkan unit ALU dapat menggeser 1 bit nilai register akumulator ke kanan. Pada pengujian ke-30, nilai akumulator 3F menjadi 9FH.

Reg A (3FH)	0	0	1	1	1	1	1	1

→ A Bit ke-0 ... Bit ke-7

Reg A (34H)	1	0	0	1	1	1	1	1

Hal ini membuktikan bahwa perintah untuk menggeser akumulator ke kanan dapat berjalan.

Pada operasi ALU untuk penggeseran bit ke kanan maupun ke kiri yang dibahas pada pengujian ke-28 hingga ke-31 ini tidak melibatkan carry. Sedangkan untuk pengujan ke-32 hingga ke-35 ini melibatkan carry. Pengujian ke-32 dan ke-33 menunjukkan ALU dapat menggeser nilai akumulator ke kiri dengan melibatkan carry. Carry adalah salah satu status hasil operasi yang terdapat dalam register flag. Untuk pengujian ke-32, carry berisi nilai 1.

Carry	Reg A (8CH)							
1	1	0	0	0	1	1	0	0

Bit ke-0 ... carry

(Cyin,A) ←

1	0	0	0	1	1	0	0	1
Carry	Reg A (19H)							

Kondisi status carry untuk hasil pada saat register A telah bernilai 19H adalah 1. Hal ini dapat dilihat pada kondisi register flag bit ke-0 pada tabel 6 no.32.

Untuk pengujian ke-33 carry diberi nilai 0.

Carry	Reg A (8CH)							
0	1	0	0	0	1	1	0	0

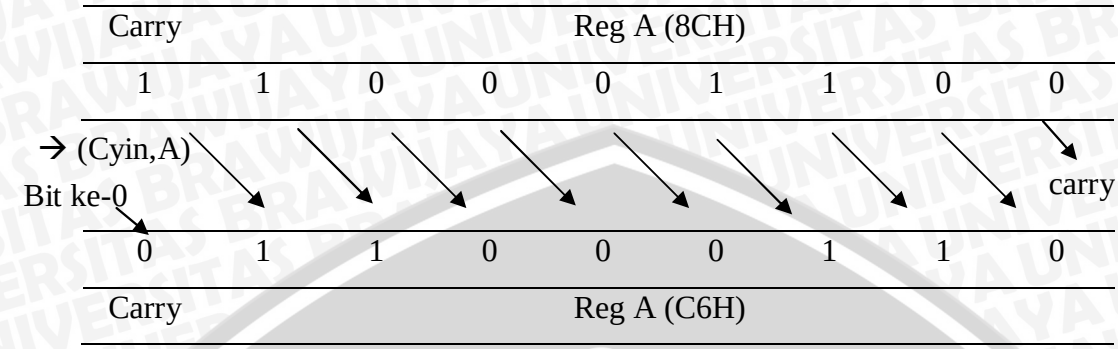
Bit ke-0 ... carry

(Cyin,A) ←

1	0	0	0	1	1	0	0	0
Carry	Reg A (18H)							

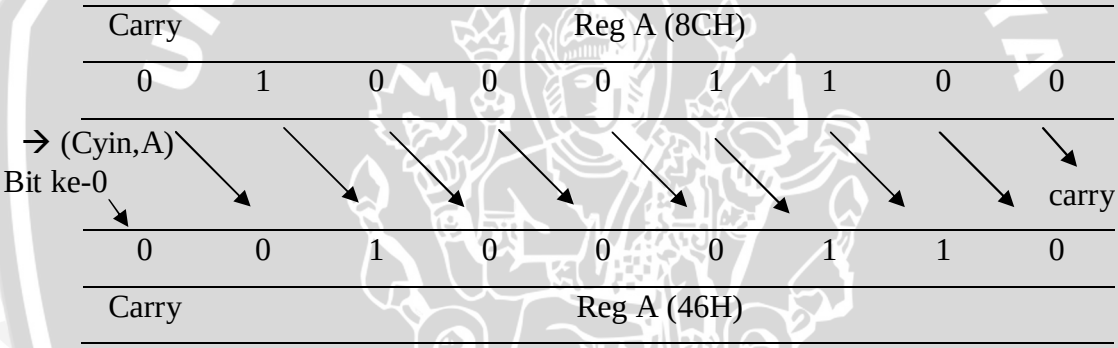
Dari pengujian ke-32 dan ke-33 unit ALU dibuktikan bahwa perintah untuk menggeser akumulator ke kiri dengan melibatkan carry dapat berjalan.

Pengujian ke-34 dan ke-35 menunjukkan ALU dapat menggeser nilai akumulator ke kanan dengan melibatkan carry. Carry adalah salah satu status hasil operasi yang terdapat dalam register flag. Untuk pengujian ke-34, carry berisi nilai 1.



Kondisi status carry untuk hasil pada saat register A telah bernilai C6H adalah 0. Hal ini dapat dilihat pada kondisi register flag bit ke-0 pada tabel 6 no.34.

Untuk pengujian ke-35 carry diberi nilai 0.



Dari pengujian ke-34 dan ke-35 unit ALU dibuktikan bahwa perintah untuk menggeser akumulator ke kanan dengan melibatkan carry dapat berjalan.

5.7.6 Analisis Hasil Pengujian Register Flag

Pada pengujian ALU, didapatkan pula hasil output untuk register flag dan dari hasil register flag tersebut terbukti bahwa status flag telah menunjukkan status yang benar dari masing-masing output ALU yang dihasilkan sesuai dengan fungsi masing-masing status. Hal ini ditunjukkan dari analisis berikut yang mengambil dari data yang didapatkan dari hasil pengujian output ALU. Data diambil secara acak dari beberapa hasil keluaran tersebut. Hasil Penjabaran Register Flag ditunjukkan dalam Tabel 22.

Tabel 23 Penjabaran Status Register Flag

No.	Output ALU	Register Flag	Status Flag				
			Sign	Zero	Aux Carry	Parity	Carry
1	01110000	00010000	0	0	1	0	0
2	10010001	10000001	1	0	0	0	1
3	11110010	10000000	1	0	0	0	0
4	00111100	00000100	0	0	0	1	0
5	10111011	10000100	1	0	0	1	0
6	01000011	00000000	0	0	0	0	0
7	10111001	10000000	1	0	0	0	0
8	10100101	10010101	1	0	1	1	1
9	01010011	00010101	0	0	1	1	1
10	10110000	10000000	1	0	0	0	0

Pada data ke-1, sign (register flag bit ke-7) bernilai 0. Hal ini dikarenakan hasil output ALU bit ke-7 bernilai 0 dan ini menandakan bahwa hasil operasi tersebut adalah bilangan positif. Hal ini berbeda dengan data ke-2. Pada data ke-2, sign bernilai 1 yang menandakan bahwa output ALU adalah bilangan negatif. Bilangan negatif juga terdapat pada output ALU data ke-3, ke-7, ke-8, dan ke-9.

Seluruh data menunjukkan bahwa tidak ada output ALU yang bernilai 0, sehingga status zero (register flag bit ke-6) bernilai 0. Jika terdapat hasil ALU yang bernilai 0 maka status zero akan bernilai 1.

Auxiliarry carry dan carry tidak dapat hanya dilihat dari hasil keluaran ALU saja karena status tersebut dapat berlogika 1 jika terdapat overflow dalam pelaksanaan instruksi. Instruksi yang dapat mempengaruhi status ini hanyalah instruksi aritmatika. Untuk instruksi logika tidak mempengaruhi hasil dua status tersebut.

Parity (register flag bit ke-2) adalah status yang menunjukkan jumlah bit yang bernilai 1 atau yang disebut parity. Jika parity genap, maka status parity bernilai 1 sedangkan jika parity-nya ganjil maka status parity bernilai 0. Pada data ke-1, parity bernilai 0 karena jumlah bit yang bernilai 1 ada 3. Sedangkan pada data ke-4, parity bernilai 1 karena jumlah bit yang bernilai 1 (parity) ada 4 (genap). Parity genap juga terdapat pada data ke-5, ke-8, dan ke-9.

5.8 Pengujian Unit Kontrol

5.8.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui apakah unit kontrol mikroprosesor 8085 dan hubungan mikroprosesor 8085 dengan blok-blok pendukung modul mikroprosesor berfungsi dengan tepat sesuai dengan spesifikasi rancangan.

5.8.2 Peralatan Pengujian

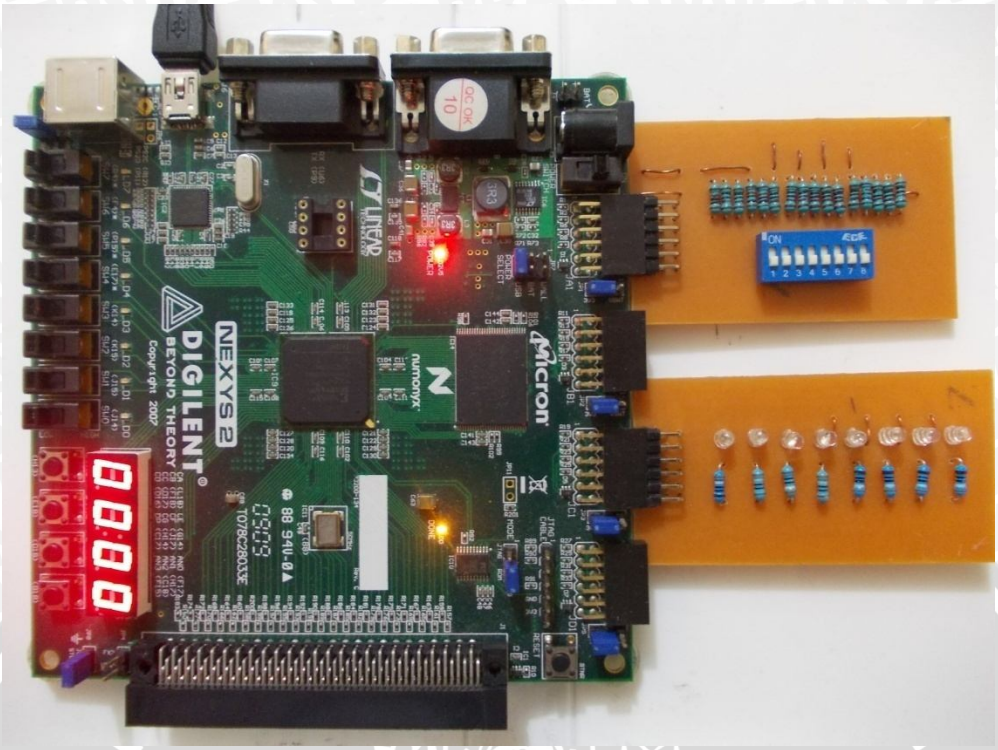
Peralatan yang digunakan dalam pengujian keseluruhan ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
4) *Software* Digilent Adept untuk antarmuka

5.8.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit memori ini adalah sebagai berikut:

1) Menyiapkan peralatan yaitu FPGA dan komputer sehingga siap untuk melakukan pemprograman.
2) Menetapkan port input yang beralamatkan 12H. Terdapat dua port output dan pemanfaatan 7-segment untuk menampilkan hasil register-register, siklus clock dan port output alamat 13H yang dikendalikan oleh switch tambahan. Perangkat pengujian ditunjukkan dalam Gambar 5.17.



Gambar 5.17 Perangkat Pengujian Unit Kontrol

3) Melakukan pengujian dengan memasukkan instruksi-instruksi ke dalam memori. Pengujian instruksi dikelompokkan sesuai dengan pengelompokan instruksi dalam bab perancangan.

a) Program 1 (Percobaan Instruksi Transfer Data)

```
MVI A,55
MVI B,32
MOV C,B
MOV A,D
MVI E,74
MVI H,21
MOV L,H
```

b) Program 2 (Percobaan Instruksi Aritmatika dan Logika)

```
MVI A,33
INR A
MVI B,02
ADD B
SUB B
RLC
INX B
```

c) Program 3 (Percobaan Instruksi Percabangan)

```
MVI A,32
INR A
JMP 0002
```

d) Program 4 (Percobaan Instruksi Input Output)

```
IN 12
OUT 13
```

4) Melihat hasil kerja unit kontrol dan mengamati perubahan pada register-register dan unit input output.

5.8.4 Data Hasil Pengujian

Pengujian dilakukan sebanyak 4 kali sesuai dengan kelompok-kelompok perintah yang kemudian dilihat perubahannya pada 7-segment maupun led indikator sebagai output. Berdasar pengujian yang dilakukan, hasil pengujian didapatkan sesuai dengan yang ditunjukkan dalam Tabel 23 hingga Tabel 26. Tabel 27 hingga Tabel 36 menunjukkan penjabaran beberapa instruksi yang digunakan pada pengujian dalam Program 1 hingga Program 4.

Tabel 24 Hasil Pengujian Program 1

No.	Alamat	Data	Assembly	Register							Port Input (12H)	Port Output (13H)
				A	B	C	D	E	H	L		
1	0000	3E	MVI A,55	00	00	00	00	00	00	00	30	00
2	0001	55		55	00	00	00	00	00	00	30	00
3	0002	06	MVI B,32	55	00	00	00	00	00	00	30	00
4	0003	32		55	32	00	00	00	00	00	30	00
5	0004	49	MOV C,B	55	32	32	00	00	00	00	30	00
6	0005	7A	MOV A,D	00	32	32	00	00	00	00	30	00
7	0006	1E	MVI E,74	00	32	32	00	00	00	00	30	00
8	0007	74		00	32	32	00	74	00	00	30	00
9	0008	26	MVI H,21	00	32	32	00	74	00	00	30	00
10	0009	21		00	32	32	00	74	21	00	30	00
11	000A	6C	MOV L,H	00	32	32	00	74	21	21	30	00

Tabel 25 Hasil Pengujian Program 2

No.	Alamat	Data	Assembly	Register							Port Input (12H)	Port Output (13H)
				A	B	C	D	E	H	L		
1	0000	3E	MVI A,33	00	00	00	00	00	00	00	30	00
2	0001	33		33	00	00	00	00	00	00	30	00
3	0002	3C	INR A	34	00	00	00	00	00	00	30	00
4	0003	06	MVI B,02	34	00	00	00	00	00	00	30	00
5	0004	02		34	02	00	00	00	00	00	30	00
6	0005	80	ADD B	36	02	00	00	00	00	00	30	00
7	0006	90	SUB B	34	02	00	00	00	00	00	30	00
8	0007	07	RLC	68	02	00	00	00	00	00	30	00
9	0008	03	INX B	68	02	01	00	00	00	00	30	00

Tabel 26 Hasil Pengujian Program 3

No.	Alamat	Data	Assembly	Register							Port Input (12H)	Port Output (13H)
				A	B	C	D	E	H	L		
1	0000	3E	MVI A,32	00	00	00	00	00	00	00	30	00
2	0001	32		32	00	00	00	00	00	00	30	00
3	0002	3C	INR A	33	00	00	00	00	00	00	30	00
4	0003	C3		33	00	00	00	00	00	00	30	00
5	0004	02	JMP 0002	33	00	00	00	00	00	00	30	00
6	0005	00		33	00	00	00	00	00	00	30	00
7	0002	3C	INR A	34	00	00	00	00	00	00	30	00

Tabel 27 Hasil Pengujian Program 4

No.	Alamat	Data	Assembly	Register							Port Input (12H)	Port Output (13H)
				A	B	C	D	E	H	L		
1	0000	DB	IN 12	00	00	00	00	00	00	00	33	00
2	0001	12		00	00	00	00	00	00	00	33	00
3	1212	33		33	00	00	00	00	00	00	33	00
4	0002	D3	OUT 13	33	00	00	00	00	00	00	33	00
5	0003	13		33	00	00	00	00	00	00	33	00
6	1313	33		33	00	00	00	00	00	00	33	33

Tabel 28 Penjabaran Siklus Clock (States) pada Instruksi MVI A,55

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M’	RD’	WR’	Reg A
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	00
	tepi turun		1	00	--	--	--	0	1	1	00
1	tepi naik		0	00	00	--	--	0	1	1	00
	tepi turun		0	00	00	--	--	0	0	1	00
2	tepi naik		0	3E	00	3E	--	0	0	1	00
	tepi turun		0	3E	00	3E	--	0	0	1	00
3	tepi naik		0	3E	00	3E	--	0	1	1	00
	tepi turun		0	3E	00	3E	--	0	1	1	00
4	tepi naik	PC ← PC+1	0	3E	00	3E	--	0	1	1	00
	tepi turun		1	01	00	3E	--	0	1	1	00
5	tepi naik		0	01	01	3E	--	0	1	1	00
	tepi turun		0	01	01	3E	--	0	0	1	00
6	tepi naik	A ← 55	0	55	01	3E	55	0	1 0	1	55
	tepi turun		0	55	01	3E	55	0	0	1	55

Tabel 29 Penjabaran Siklus Clock (States) pada Instruksi MOV C,B

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M’	RD’	WR’	Reg B	Reg C
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	32	00
	tepi turun		1	04	--	--	--	0	1	1	32	00
1	tepi naik		0	04	04	--	--	0	1	1	32	00
	tepi turun		0	04	04	--	--	0	0	1	32	00
2	tepi naik		0	49	04	49	--	0	0	1	32	00
	tepi turun		0	49	04	49	--	0	0	1	32	00
3	tepi naik	C ← B	0	49	04	49	--	0	1	1	32	32
	tepi turun		0	49	04	49	--	0	1	1	32	32

Tabel 30 Penjabaran Siklus Clock (States) pada Instruksi INR A

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M’	RD’	WR’	Reg A
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	33
	tepi turun		1	02	--	--	--	0	1	1	33
1	tepi naik		0	02	02	--	--	0	1	1	33
	tepi turun		0	02	02	--	--	0	0	1	33
2	tepi naik	in_alu ← 00001	0	3C	02	3C	--	0	0	1	33
	tepi turun		0	3C	02	3C	--	0	0	1	33
3	tepi naik	A ← out_alu	0	3C	02	3C	--	0	1	1	34
	tepi turun		0	3C	02	3C	--	0	1	1	34

Tabel 31 Penjabaran Siklus Clock (States) pada Instruksi ADD B

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M’	RD’	WR’	Reg A	Reg B
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	34	02
	tepi turun		1	05	--	--	--	0	1	1	34	02
1	tepi naik		0	05	05	--	--	0	1	1	34	02
	tepi turun		0	05	05	--	--	0	0	1	34	02
2	tepi naik	Temp_Reg ← B in_alu ← 00010	0	80	05	80	--	0	0	1	34	02
	tepi turun		0	80	05	80	--	0	0	1	34	02
3	tepi naik	A ← out_alu	0	80	05	80	--	0	1	1	36	02
	tepi turun		0	80	05	80	--	0	1	1	36	02

Tabel 32 Penjabaran Siklus Clock (States) pada Instruksi SUB B

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M'	RD'	WR'	Reg A	Reg B
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	36	02
	tepi turun		1	06	--	--	--	0	1	1	36	02
1	tepi naik		0	06	06	--	--	0	1	1	36	02
	tepi turun		0	06	06	--	--	0	0	1	36	02
2	tepi naik	Temp_Reg ← B in_alu ← 00000	0	90	06	90	--	0	0	1	36	02
	tepi turun		0	90	06	90	--	0	0	1	36	02
3	tepi naik	A ← out_alu	0	90	06	90	--	0	1	1	34	02
	tepi turun		0	90	06	90	--	0	1	1	34	02

Tabel 33 Penjabaran Siklus Clock (States) pada Instruksi RLC

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M'	RD'	WR'	Reg A
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	34
	tepi turun		1	07	--	--	--	0	1	1	34
1	tepi naik		0	07	07	--	--	0	1	1	34
	tepi turun		0	07	07	--	--	0	0	1	34
2	tepi naik	in_alu ← 10101	0	07	07	07	--	0	0	1	34
	tepi turun		0	07	07	07	--	0	0	1	34
3	tepi naik	A ← out_alu	0	07	07	07	--	0	1	1	68
	tepi turun		0	07	07	07	--	0	1	1	68

Tabel 34 Penjabaran Siklus Clock (States) pada Instruksi INX B

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M'	RD'	WR'	Reg B	Reg C
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	02	00
	tepi turun		1	08	--	--	--	0	1	1	02	00
1	tepi naik		0	08	08	--	--	0	1	1	02	00
	tepi turun		0	08	08	--	--	0	0	1	02	00
2	tepi naik	Temp_reg ← C in_alu ← 00100	0	03	08	03	--	0	0	1	02	00
	tepi turun		0	03	08	03	--	0	0	1	02	00
3	tepi naik	C ← out_alu	0	03	08	03	--	0	1	1	02	01
	tepi turun		0	03	08	03	--	0	1	1	02	01
4	tepi naik	if (CY==1) => temp_reg ← B in_alu ← 00100	0	03		03	--	0	1	1	02	01
	tepi turun		0	03		03	--	0	1	1	02	01
5	tepi naik	if (CY==1) => B ← out_alu,, if (CY==0) => B ← B	0	03		03	--	0	1	1	02	01
	tepi turun		0	03		03	--	0	1	1	02	01

Tabel 35 Siklus Clock (States) pada Instruksi JMP 0002

Siklus	Pemicuan	operasi	ALE	AD (μP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand1	Operand0	IO/M'	RD'	WR'
0	tepi naik	PC ← PC+1	0	--	--	--	--	--	0	1	1
	tepi turun		1	03	--	--	--	--	0	1	1
1	tepi naik		0	03	03	--	--	--	0	1	1
	tepi turun		0	03	03	--	--	--	0	0	1
2	tepi naik		0	C3	03	C3	--	--	0	0	1
	tepi turun		0	C3	03	C3	--	--	0	0	1
3	tepi naik		0	C3	03	C3	--	--	0	1	1
	tepi turun		0	C3	03	C3	--	--	0	1	1
4	tepi naik	PC ← PC+1	0	C3	03	C3	--	--	0	1	1
	tepi turun		1	04	03	C3	--	--	0	1	1
5	tepi naik		0	04	04	C3	--	--	0	1	1
	tepi turun		0	04	04	C3	--	--	0	0	1
6	tepi naik		0	02	04	C3	--	02	0	0	1
	tepi turun		0	02	04	C3	--	02	0	0	1
7	tepi naik	PC ← PC + 1	0	02	04	C3	--	02	0	1	1
	tepi turun		1	05	04	C3	--	02	0	1	1
8	tepi naik		0	05	05	C3	--	02	0	1	1
	tepi turun		0	05	05	C3	--	02	0	0	1
9	tepi naik		0	00	05	C3	00	02	0	0	1
	tepi turun		0	00	05	C3	00	02	0	0	1
		PC ← 0002									

Tabel 36 Penjabaran Siklus Clock (States) pada Instruksi IN 12

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M’	RD’	WR’	Reg A
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	00
	tepi turun		1	00	--	--	--	0	1	1	00
1	tepi naik		0	00	00	--	--	0	1	1	00
	tepi turun		0	00	00	--	--	0	0	1	00
2	tepi naik		0	DB	00	DB	--	0	0	1	00
	tepi turun		0	DB	00	DB	--	0	0	1	00
3	tepi naik		0	DB	00	DB	--	0	1	1	00
	tepi turun		0	DB	00	DB	--	0	1	1	00
4	tepi naik	PC ← PC+1	0	DB	00	DB	--	0	1	1	00
	tepi turun		1	01	00	DB	--	0	1	1	00
5	tepi naik		0	01	01	DB	--	0	1	1	00
	tepi turun		0	01	01	DB	--	0	0	1	00
6	tepi naik		0	12	01	DB	12	0	0	1	00
	tepi turun		0	12	01	DB	12	0	0	1	00
7	tepi naik		0	12	01	DB	12	0	1	1	00
	tepi turun		1	12	01	DB	12	1	1	1	00
8	tepi naik		0	12	12	DB	12	1	1	1	00
	tepi turun		0	12	12	DB	12	1	0	1	00
9	tepi naik	A ← input	0	33	12	DB	12	1	0	1	33
	tepi turun		0	33	12	DB	12	1	0	1	33

Tabel 37 Penjabaran Siklus Clock (States) pada Instruksi OUT 13

Siklus	Pemicuan	operasi	ALE	AD (µP 8085)	Alamat 0:7 (out_bidirect)	Reg_Instr	Operand0	IO/M'	RD'	WR'	Output
0	tepi naik	PC ← PC+1	0	--	--	--	--	0	1	1	00
	tepi turun		1	02	--	--	--	0	1	1	00
1	tepi naik		0	02	02	--	--	0	1	1	00
	tepi turun		0	02	02	--	--	0	0	1	00
2	tepi naik		0	D3	02	D3	--	0	0	1	00
	tepi turun		0	D3	02	D3	--	0	0	1	00
3	tepi naik		0	D3	02	D3	--	0	1	1	00
	tepi turun		0	D3	03	D3	--	0	1	1	00
4	tepi naik	PC ← PC+1	0	D3	02	D3	--	0	1	1	00
	tepi turun		1	03	02	D3	--	0	1	1	00
5	tepi naik		0	03	03	D3	--	0	1	1	00
	tepi turun		0	03	03	D3	--	0	0	1	00
6	tepi naik		0	13	03	D3	13	0	0	1	00
	tepi turun		0	13	03	D3	13	0	0	1	00
7	tepi naik		0	13	03	D3	13	0	1	1	00
	tepi turun		1	13	03	D3	13	1	1	1	00
8	tepi naik		0	13	13	D3	13	1	1	1	00
	tepi turun		0	13	13	D3	13	1	1	0	00
9	tepi naik	Output ← A	0	33	13	D3	13	1	1	0	33
	tepi turun		0	33	13	D3	13	1	1	0	33

5.8.5 Analisis Hasil Pengujian

Pada pengujian keseluruhan, unit kontrol telah dapat menjalankan instruksi yang diberikan dan blok-blok yang terintegrasi secara menyeluruh telah berfungsi sesuai dengan spesifikasi perancangan. Hal ini ditunjukkan dari hasil transfer data yang telah sesuai dengan instruksi yang diberikan. Hasil operasi aritmatika logika terbukti telah sesuai dengan instruksi dan juga perhitungan. Instruksi percabangandan instruksi memasukkan data dari unit input maupun menampilkan data ke unit output juga telah berjalan sesuai dengan yang diinginkan.

Sebagai contoh adalah program 2. Pada program ini, unit kontrol menjalankan perintah yaitu:

1) MVI A,33 yaitu akumulator akan diberikan nilai 33H. Perintah tersebut dilaksanakan dengan bukti yang tampak pada display bahwa isi akumulator menjadi angka 33H.
2) INR A yaitu nilai akumulator ditambahkan 1. Perintah ini berhasil dilaksanakan dengan berubahnya display yang menunjukkan nilai register akumulator dari 33H menjadi 34H.
3) MVI B,02 yaitu akumulator akan diberikan nilai 02H. Perintah tersebut dilaksanakan dengan bukti yang tampak pada display bahwa isi register B menjadi angka 02H.
4) ADD B adalah instruksi untuk menambahkan angka isi register A dengan angka isi register B, dan hasilnya disimpan dalam register A. Perintah tersebut dilaksanakan dengan bukti yang tampak pada display bahwa isi akumulator menjadi angka 36H yang berarti 34H+02H=36H.
5) SUB B adalah instruksi untuk mengurangi angka isi register A dengan angka isi register B, dan hasilnya disimpan dalam register A. Perintah tersebut dilaksanakan dengan bukti yang tampak pada display bahwa isi akumulator menjadi angka 34H yang berarti 36H-02H=36H.
6) RLC adalah instruksi yang memerintahkan untuk menggeser nilai register A 1 bit ke kiri. Perintah tersebut dilaksanakan dengan bukti yang tampak pada display bahwa nilai 34H menjadi 68H atau dapat terlihat jika dikonversi ke dalam biner, 00110100b menjadi 01101000b.

7) INX B adalah instruksi yang memerintahkan untuk menambahkan nilai 1 ke register pair B. Pasangan register B adalah C. Nilai awal register C adalah 00H berubah menjadi 01H dengan adanya instruksi ini.

5.9 Komparasi Modul Mikroprosesor 8085 dengan Sistem

5.9.1 Tujuan Pengujian

Pengujian ini dilakukan guna mengetahui perbandingan sistem yang dirancang dengan modul mikroprosesor 8085 yang telah digunakan untuk praktikum apakah sistem telah dapat menggantikan modul yang telah ada. Modul mikroprosesor menggunakan frekuensi tipikal mikroprosesor sebesar 2 MHz sedangkan sistem menggunakan frekuensi maksimum yaitu sebesar 3,25 MHz.

5.9.2 Peralatan Pengujian

Peralatan yang digunakan dalam pengujian unit *input output* ini antara lain adalah:

1) FPGA XILINX 3E-500 FG320
2) Komputer dengan dukungan USB 2.0 dengan daya 500 mA
3) Logic Analyzer
4) *Software* Xilinx ISE Project Navigator 11.1 untuk pemprograman
5) *Software* Digilent Adept untuk antarmuka
6) Software ELAB-080 untuk mengetahui sinyal
7) ModulKit Praktikum Mikroprosesor 8085

5.9.3 Prosedur Pengujian

Prosedur dalam melakukan pengujian unit *clock* ini adalah sebagai berikut:

1) Menyiapkan peralatan yaitu FPGA, komputer, dan logic analyzer sehingga siap untuk melakukan pemprograman.
2) Memuat program untuk pengujian ke dalam FPGA sebagai berikut.

```
IN 12
MOV B,A
INR B
MOV A,B
OUT 13
JMP 0000
```

3) Menyambungkan port yang ditentukan ke logic analyzer. Port-port yang akan disambungkan pada logic analyzer adalah clock, ALE, IO/M', RD',WR, AD0, dan A0.
4) Menyambungkan *USB connection* logic analyzer ke komputer.
5) Mengulangi prosedur ke-2 sampai ke-4 namun dalam ModulKit Pratikum Mikroprosesor 8085. Penyusunan perangkat ditunjukkan dalam Gambar 5.18

Gambar 5. 18 Perangkat Pengujian Menggunakan ModulKit

6) Sedangkan penyusunan perangkat dengan sistem yang dirancang ditunjukkan dalam Gambar 5.19

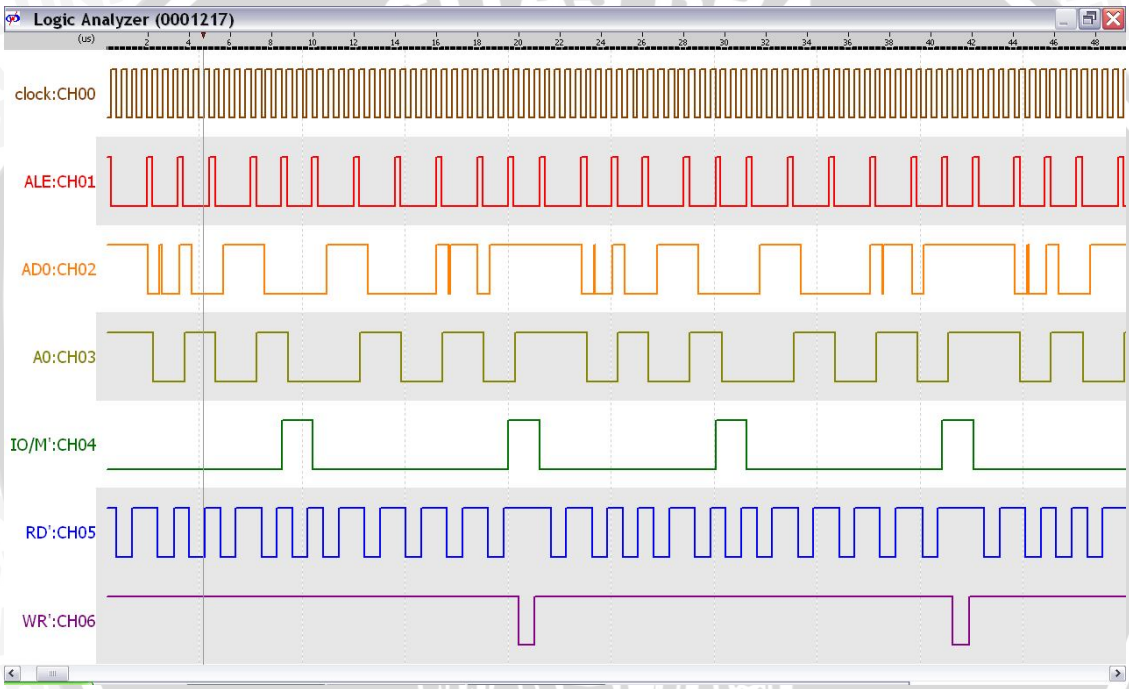
Gambar 5. 19 Perangkat Pengujian Menggunakan FPGA

7) Melihat hasil sinyal dalam software ELAB-080.

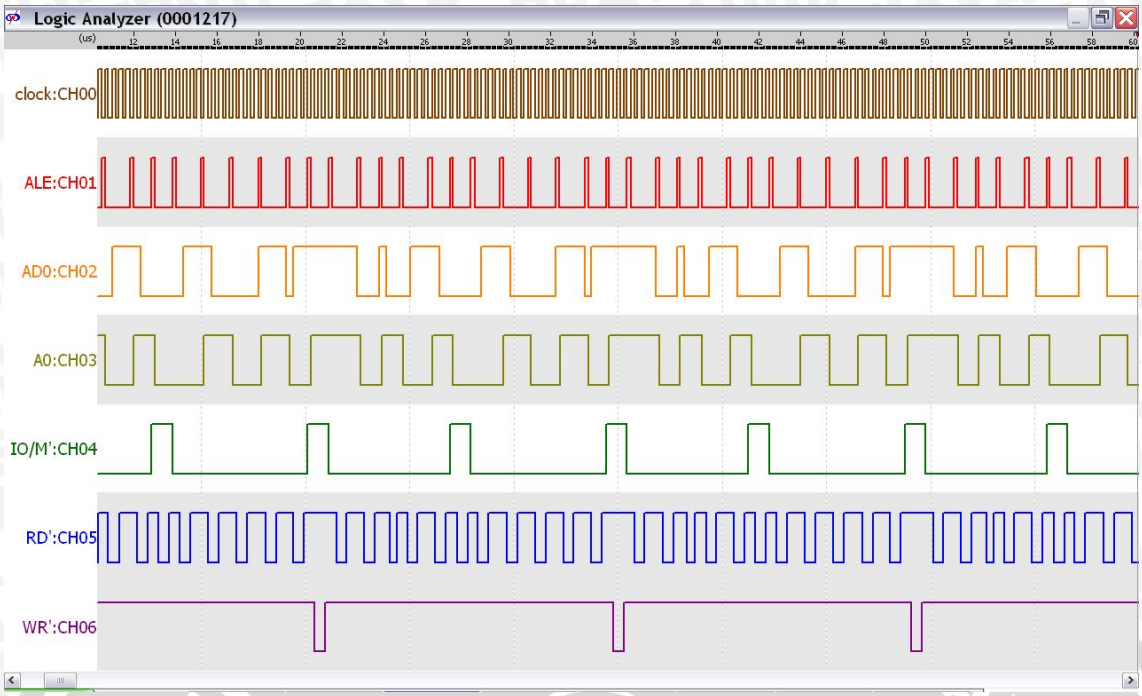
5.9.4 Data Hasil Pengujian

Data yang didapat saat pengujian adalah sinyal clock, ALE, AD0, A0, IO/M', RD', dan WR'. Sinyal clock, ALE,AD0, IO/M', RD', dan WR' adalah sinyal-sinyal keluaran mikroprosesor sedangkan A0 adalah sinyal keluaran dari unit bidirectional, yaitu unit yang menghubungkan mikroprosesor dengan unit luar seperti memori ataupun unit input output. A0 adalah port yang menunjukkan alamat bit ke-0.

Data hasil pengujian saat menggunakan modulkit praktikum mikroprosesor 8085 ditunjukkan dalam Gambar 5.20 sedangkan hasil pengujian menggunakan implementasi FPGA ditunjukkan dalam Gambar 5.21.



Gambar 5. 20 Hasil Pengujian Saat Menggunakan Modulkit Praktikum Mikroprosesor 8085



Gambar 5. 21 Hasil Pengujian Saat Menggunakan Rancangan Sistem dalam FPGA

5.9.1 Analisis Hasil Pengujian

Berdasar data yang didapatkan dari hasil komparasi antara sinyal-sinyal keluaran dari pengujian menggunakan modul praktikum yang terdapat di Laboratorium Sistem Digital TEUB dengan sinyal keluaran FPGA terlihat bahwa keluaran sinyal-sinyal sama dan sesuai dengan picu clock masing-masing (untuk modul praktikum menggunakan frekuensi tipikal sebesar 2 MHz dan untuk mikroprosesor 8085 dalam FPGA menggunakan frekuensi maksimum mikroprosesor 8085 yaitu 3,25 MHz).

ALE (*Address Latch Enable*) adalah sinyal yang mengaktifkan latch alamat. Sistem latch digunakan pada unit buffer dan unit bidirectional sehingga saat ALE telah aktif maka mikroprosesor 8085 telah dapat mengirimkan alamat ke unit luar. ALE membutuhkan waktu 0,5 clock.

AD0 adalah representasi dari port address data pada mikroprosesor dan yang ditampilkan dalam diagram waktu ini adalah port address data bit ke-0. Port ini mengirimkan alamat bit ke-0 sampai ke-7 yang dikirimkan mikroprosesor ke luar atau data (8 bit) yang masuk ke mikroprosesor maupun keluar dari mikroprosesor. Saat ALE aktif, port AD akan mengirimkan alamat sedangkan saat ALE tidak aktif, port AD akan menggambarkan nilai data yang akan terakses maupun yg diakses oleh mikroprosesor (akan berubah menjadi nilai data kembali 1 clock setelah tepi turun ALE).

A0 adalah representasi dari port address keluaran unit bidirectional dan akan aktif setelah ALE aktif (saat ALE *falling-edge*) dan yang ditampilkan dalam diagram waktu ini adalah port address data bit ke-0.

IO/M' merepresentasikan sinyal IO/M' mikroprosesor 8085. Saat IO/M' berlogika rendah maka yang akan diakses oleh mikroprosesor adalah memori sedangkan saat IO/M' berlogika tinggi maka yang akan diakses oleh mikroprosesor adalah unit input atau unit output.

RD' merepresentasikan sinyal RD' mikroprosesor 8085. Saat RD' berlogika rendah maka mikroprosesor akan membaca data dari unit luar dan unit luar tersebut memori atau input tergantung dari sinyal IO/M'. Dalam siklus pembacaan, sinyal RD' aktif selama 1,5 clock.

WR' merepresentasikan sinyal WR' mikroprosesor 8085. Saat WR' berlogika rendah maka mikroprosesor akan menuliskan data ke unit luar dan unit luar tersebut memori atau input tergantung dari sinyal IO/M'. Dalam siklus penulisan, sinyal WR' aktif selama 1,5 clock.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan pengujian dan analisis keseluruhan yang telah dijabarkan dalam bab V, dapat ditarik kesimpulan sebagai berikut:

1) Sistem mikroprosesor 8085 dalam FPGA adalah suatu mikroprosesor yang dapat berfungsi jika memiliki program counter 16 bit, ALU 8 bit, register 8 bit yaitu A, B, C, D, E, H, L, register instruksi, dekoder instruksi, port address data 8 bit dan port address 8 bit, serta unit kontrol yang mengirimkan sinyal ALE, IO/M', RD', dan WR'.
2) Frekuensi sistem clock dari sistem yang dirancang lebih tinggi yaitu sebesar 3,25 MHz jika dibandingkanfrekuensi modulkit mikroprosesor 8085 yang digunakan di Laboratorium Sistem Digital TEUB yaitu sebesar 2 Mhz sehingga pemrosesan instruksi berjalan lebih cepat.
3) Sistem memori yang dirancang dalam FPGA dapat kompatibel dengan mikroprosesor 8085 dengen spesifikasi memori berkapasitas 256 bit dengan jalur data 8 bit dan jalur alamat 16 bit.
4) Adanya port address dan data yang tergabung dalam sistem mikroprosesor 8085 membutuhkan suatu unit yang mengolah data tersebut agar data dan alamat dapat terpisah yaitu unit bidirectional. Unit buffer dibutuhkan untuk mengontrol alamat (8:15) dari mikroprosesor ke unit luar dan unit dekoder digunakan untuk mengubah sinyal kontrol dari mikroprosesor menjadi sinyal sesuai dengan kontrol untuk masing-masing unit luar yaitu IO/M', RD', dan WR' menjadi MEMR', MEMW', IOR', dan IOW'.
5) Sistem yang telah dirancang telah dapat melakukan fungsi-fungsi dan perintah-perintah sesuai dengan instruksi yang dijalankan serta menampilkan diagram waktu yang serupa dengan diagram waktu yang dihasilkan oleh modulkit mikroprosesor 8085.

6.2 Saran

Saran yang dapat diberikan dalam perancangan dan pembuatan sistem mikroprosesor 8085 dan memori adalah sebagai berikut.

1) Untuk penggunaaan praktikum, set instruksi telah mencukupi namun dibutuhkan set intruksi lebih banyak lagi agar sistem ini dapat digunakan dalam kawasan yang lebih luas lagi.
2) Disarankan untuk menambahkan *stack pointer* dan *interrupt*.
3) Pada sistem ini dapat ditambahkan perangkat monitor dan keyboard serta program tambahan sebagai driver maupun assembler untuk instruksi agar memasukkan instruksi-instruksi ataupun perintah-perintah mikroprosesor dapat melalui perangkat tersebut dan juga dapat langsung menggunakan bahasa assembly.

DAFTAR PUSTAKA

Chu, Pong P. 2008. FPGA Prototyping by VHDL Examples Xilinx Spartan™-3 Version.New Jersey:John Wiley & Sons, Inc.

Dubey,Rahul. 2004. Introduction to Embedded System Design Using Field Programmable Gate Arrays. New York: Springer.

Hartenstein, R.W. 1994. Field Programmable Logic Architectures, Synthesis and Applications. New York: Springer.

Hery_h.staff.gunadarma.ac.id/…/Materi Perkembangan 1 Mikroprosesor.PDF, tanggal akses 7 Februari 2011.

Maxfield, Clive. 2009. FPGAs World Class Design. San Fransisco: Newnes.

Lee, Weng Fuok. 2000. VHDL Coding and Logic Synthesis with Synopsis. San Diego:Academic Press.

Roth, Charles H.,Jr. 1998. Digital Systems Design Using VHDL. Boston:PWS Publishing Company.

Scribd. Mikroprosesor.*www.scribd.com/doc/ 14684664/* ***mikroprosesor*** , diakses tanggal 5 Maret 2011.

Sen, S.K. 2010. Understanding 8085/8086 Microprosessors and Peripheral ICs Through Questions and Answers. New Delhi:New Age International (P) Limited, Publishers.

Thomas, Donald E., Philip R. Moorby. 2002. The Verilog Hardware Description Language, Fifth Edition.New York:Kluwer Academic Publisher.

Webphysics.davidson. Microprosesor 8085.w*ebphysics.davidson.edu/ faculty/dmb/ py310/****8085,*** *diakses tanggal 5 Maret 2011*

LAMPIRAN
LISTING PROGRAM VHDL