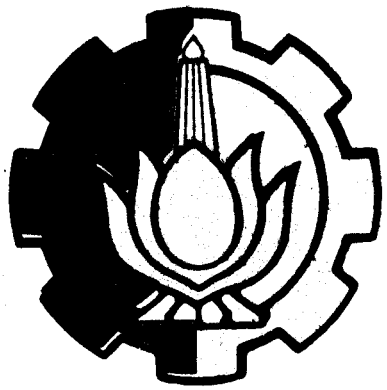
PERENCANAAN DAN PEMBUATAN UNIT PEMROSES KOMPUTER DIGITAL DELAPAN BIT



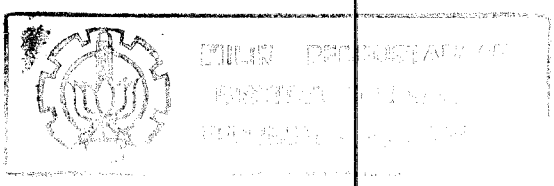
OLEH :

ABDUL BASITH

NRP : 2842200200

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
S U R A B A Y A

1993



PERENCANAAN DAN PEMBUATAN UNIT PEMROSES KOMPUTER DIGITAL DELAPAN BIT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar
Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui
Dosen Pembimbing

Ir. HARMANI SOEHARDJO

SURABAYA
MARET, 1993

Abstrak

Kebutuhan akan mikroprosessor baik sebagai kontrol maupun sebagai komputer yang semakin meningkat mendorong para perancang berlomba-lomba untuk meningkatkan kemampuan serta keandalan mikroprosessor.

Untuk mengembangkan mikroprosessor diperlukan pengetahuan dasar tentang rangkaian penyusun dan sistem kerja mikroprosessor.

Pada tugas akhir ini dicoba untuk merencakan dan membuat unit pemroses komputer digital delapan bit yang dibuat secara diskrit dari IC-IC transistor-transistor logic (TTL) yang dapat meniru system kerja mikroprosessor Z80.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmaNya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul:

PERENCANAAN DAN PEMBUATAN
UNIT PEMROSES KOMPUTER DIGITAL DELAPAN BIT

Yang mempunyai beban 6 SKS (Satuan Kredit Semester) dan pelengkap persyaratan guna memperoleh gelar sarjana pada jurusan teknik Elektro, Fakultas Teknologi Industri Institut teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan tugas akhir ini, penulis melakukan penyusunan berdasarkan pada teori-teori yang sudah pernah diperoleh selama masih kuliah, berbagai literatur-literatur ilmiah, bimbingan dari dosen pembimbing serta pihak-pihak lain yang telah mendorong semangat penyusunan.

Penulis berharap agar tugas akhir ini dapat memberikan manfaat bagi yang memerlukannya.

Tak lupa pula penulis sampaikan banyak terima kasih atas bimbingan, nasihat serta dorongan dari:

pembimbing tugas akhir.

2. Ir. Soetikno selaku koordinator bidang studi Elektronika.
3. Ir. Katjuk Astrowulan MSEE selaku ketua Jurusan Teknik Elektro FTI ITS.
4. Adik-kakakku, rekan-rekan mahasiswa bidang studi elektronika, serta banyak pihak yang tak dapat penulis sebutkan satu persatu, yang telah banyak membantu dalam penyelesaian tugas akhir ini.

Surabaya, Pebruari 1992

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

I.1 LATAR BELAKANG

Penggunaan komputer disegala bidang banyak dijumpai dalam kehidupan sehari-hari, baik untuk keperluan kantor, pabrik sebagai pengontrol alat / mesin atau untuk keperluan pribadi dalam berbagai keperluan, maka hampir semua kebutuhan modern memerlukan komputer.

Unjuk kerja komputer tidak terlepas dari prosessor yang selanjutnya biasa disebut mikroprosessor. Dimana di dalam prosessor tersebut semua proses dilakukan baik berupa perhitugan, proses logika dan pengontrolan terhadap jalannya data.

Secara umum kerja berbagaai jenis prosessor maupun karakteristiknya dapat dipelajari dari buku manual dari pabriknya masing-masing. Namun bagaimana rangkaian dalam tiap-tiap mikroprosessor tidak satupun pabrik yang memberitahukannya karena hal ini adalah rahasia bagi pabrik-pabrik tersebut.

Dalam tugas akhir ini akan dicoba untuk merangkai dari IC - IC TTL untuk membuat prrosessor digital 8 bit yang akan meniru sistem kerja mikroprosessor yang telah ada, dimana disini yang akan dicoba untuk ditiru adalah mikroprosessor Z80.

I.2 PERMASALAHAN

Pembuatan mikroprosessor dilakukan dalam rangkaian *Integrated* yang berskala besar (VLSI) sehingga bentuknya sangat kecil utuk beribu-ribu bahkan berjuta-juta transistor didalamnya. Sedangkan untuk mempelajari rangkaian dalamnya bahkan sistem kerja rangkaian, tentunya dipelukan tidak dalam bentuk IC, melainkan perlu dirangkai secara diskrit penyusunan-penyusunannya melalui bread board yang banyak. Sehingga dapat dengan mudah diadakan trial and error.

I.3 PEMBATASAN MASALAH

Dalam meniru sistem kerja Z80 akan diperlukan rangkaian yang sangat banyak. Dalam perencanaan ini akan dibuat sebagian perintah - perintah Z80 yang dapat menampilkan simulasi huruf berjalan pada led matriks 8 x 8. Dimana perintah - perintah tersebut meliputi penyalinan (loading) data dari register ke register, register ke memory, pemindahan data dari blok memory ke blok memory yang lain, pergeseran data dalam register, compare data yang menghasilkan zero flag dimana flag ini akan berguna pada instruksi jump yang bersyarat dan pada perintah move blok yaitu dipakai sebagai pendeteksi counter registernya.

Dalam pembuatan rangkaian ini yang dipentingkan adalah perujudan logikanya dan tidak dipentingkan perhitungan kecepatan kerja alat, dan daya listrik yang dipelukan oleh alat tersebut.

I.4 METODOLOGI

Dibuat rangkaian yang bisa mengambil kode-kode operasi (bahasa mesin) Z80 yang terdapat pada memory external dan dibuat rangkaian yang bisa menerjemahkan kode-kode tersebut menjadi perintah yang sesuai.

I.5 LANGKAH PEMBAHASAN

Setelah mengulas tentang bab pendahuluan, pada bab dua akan dibahas segala sesuatu mengenai teori dasar pemroses digital. Pada bab tiga akan dibahas bagaimana cara menerjemahkan kode - kode operasi ke dalam rangkaian, sedangkan pada bab empat akan dibahas mengenai hasil pengukuran dan yang terakhir yaitu bab ke lima akan dibuat kesimpulan dari seluruh bab yang ada.

BAB II
TEORI PENUNJANG

II.1 PENDAHULUAN

Pada bab ini akan dibahas teori penunjang yang berhubungan dengan alat yang dibuat, pertama akan dibahas teori tentang komputer, kemudian mengenai CPU dan instruksi-instruksi Z80.

II.2 PENGERTIAN KOMPUTER[1)]

Kata komputer meliputi banyak jenis mesin, secara luas perbedaan dalam ukuran kecepatan dan harga. Ini dapat dipakai untuk menggambarkan beberapa subklas dari pada komputer-komputer. Komputer yang paling kecil biasanya disebut minikomputer, yang mewakili secara relatip yang paling rendah harganya, ukurannya dan kemampuan penghitungannya. Pada awal tahun 1970 istilah mikrokomputer telah dikenalkan untuk menggambarkan sebuah komputer yang sangat kecil, rendah dalam harga dan hanya tersusun dari sejumlah kecil kemasan "skala besar rangkaian terintegrasi" (VSLI).

Komputer-komputer besar biasanya disebut "mainframe" yang jelas / sama sekali berbeda dengan mikrokomputer-mikrokomputer dalam ukuran, kekuatan pemrosesan, harga dan kekomplekkannya maupun pengalaman dari desainnya.

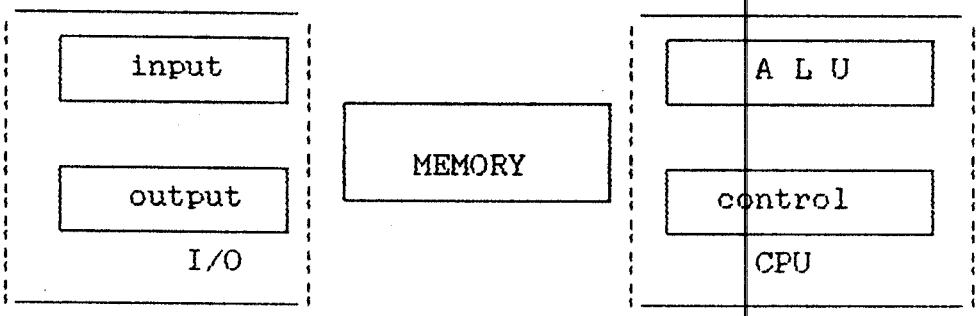
---

1) V. Carl Hamacher, Zvonsko G. Vranesic, Safwat G. zaky Computer Organization, McGraw-Hill International Editions, 1984, hal. 1.

Akan tetapi konsep -konsep dasar secara essential untuk semua kelas komputer adalah sama.

II.3. BAGIAN-BAGIAN PENYUSUN KOMPUTER

Dalam bentuk yang paling sederhana, sebuah komputer tersusun dari lima bagian fungsi pokok meliputi: input, memory, aritmatik dan logika, output dan unit-unit kontrol seperti terlihat pada gambar 2.1



GAMBAR 2.1[2)]

DASAR UNIT-UNIT FUNGSIONAL SEBUAH KOMPUTER

Unit input menerima informasi yang dikodekan dari seorang operator, dari peralatan elektromekanis, atau menerima dari hubungan komputer - komputer lain melalui jalur-jalur komunikasi data. Informasi juga disimpan pada memory sebagai referensi atau secara langsung dijalankan oleh rangkaian ALU yang membentuk operasi-operasi yang dikehendaki. Langkah-langkah pemrosesan ditentukan oleh program yang disimpam di memory. Akhirnya hasil dikirim kembali ke bagian luar melalui unit output. Semua

---

2)Ibid hal.2

kegiatan dikoordinasikan oleh kontrol unit. Diagram pada gambar 2.1 memperlihatkan hubungan-hubungan antar unit fungsi. Ada beberapa cara yang dipakai dalam hubungan-hubungan komputer, yang tak akan dibahas dalam buku ini akan tetapi secara umum akan menunjukkan pada sebuah komputer tunggal yang menggambarkan prinsip - prinsip dasar, banyak sistem komputer modern meliputi sejumlah interkoneksi antar komputer yang disebut jaringan komunikasi (network comunication) . Istilah "distributed computing" sering dipakai untuk menunjukkan lingkungan yang demikian. Sudah menjadi kebiasaan menyebut pada rangkaian Aritmatik dan logika (ALU) dalam hubungan dengan rangkaian kontrol utama sebagai "Control Processing Unit " (CPU), atau secara sederhana prosesor. Kata kontrol dipakai untuk menunjukkan sebagian besar fungsi-fungsi kontrol yang diberikan komputer dipusatkan pada unit tunggal. Sistem modern sering mengandung banyak prosesor, masing-masing menunjukkan untuk membentuk fungsi secara perbagian. Perlengkapan input dan output biasanya digabungkan dalam istilah "input - output unit" (I/O). Dalam kenyataan biasanya beberapa peralatan standard menyajikan fungsi - fungsi input dan output sekaligus. Contoh yang paling sederhana, sering terminal "Encountered Cathode Ray tube" (CRT), terdiri dari sebuah keyboard sebagai input dan display CRT sebagai output. Fungsi - fungsi input dan output harus diabaikan dalam terminal yang dipisahkan.

Dalam komputer-komputer yang besar unit-unit fungsi utama dapat berisi sejumlah bagian dan sering berukuran

besar secara fisik. Minikomputer dan mikrokomputer berukuran lebih kecil, sering berdimensi meja.

Pada poin ini akan ditutup dengan pembicaraan tentang bentuk informasi dalam komputer. Informasi ini akan dibedakan dalam dua type, yang disebut instruksi dan data. Instruksi adalah komando - komando yang:

- Memerintah transfer informasi dalam komputer, seperti antara komputer dan I/O devicee.
- Mentransfer operasi-operasi aritmatik dan algoritma untuk dibentuk.

Seperangkat instruksi yang membentuk sebuah tugas disebut " program ". Biasanya disimpan dalam bentuk program (beberapa program) dalam memory. Kemudian prosesor mengambil instruksi-instruksi dalam program dari memory dan membentuk operasi-operasi yang diinginkan. Instruksi-instruksi secara normal dijalankan secara urut menurut urutan penyimpanan. Meskipun mungkin menyimpang dari order seperti keadaan percabangan dibutuhkan. Kebiasaan aktual pada sebuah komputer di bawah kontrol komplit pada program simpanan. Kecuali interupsi dari luar oleh operator atau rangkaian digital yang dihubungkan ke mesin.

Data adalah angka-angka dan karakter-karakter yang dipakai sebagai operand-operand oleh instruksi. Ini tidak akan diinterprestasikan sebagai definisi keras, akan tetapi istilah ini sering dipakai sebagai simbol beberapa informasi digital. Sama dalam mendefinisikan pada data, bahwa program masukan dapat diputuskan sebagai data jika

diproses dengan program lain. Sebagai contoh proses kompilasi pada sumber program dari bahasa tingkat tinggi ke dalam instruksi-instruksi dan data bahasa mesin. Program sumber adalah input data bagi program compiler, compiler mengubah program sumber ke dalam program bahasa mesin.

Informasi yang dihandalkan oleh komputer harus diterjemahkan dengan format yang sesuai. Karena Hardware saat ini menggunakan rangkaian digital yang hanya punya dua keadaan secara alami yang dinamakan ON dan OFF, pengkodean secara biner digunakan. Ini berarti tiap-tiap angka, karakter teks, atau instruksi dikodekan sebagai string digit-digit biner yang masing-masing hanya punya satu dari dua nilai kemungkinan. Angka-angka biasanya diwakili dalam notasi posisi biner. Karakter-karakter alfa numerik biasanya diekspresikan dalam sebutan-sebutan kode biner. Beberapa skema kode telah dibuat. Dua yang umum dipakai adalah ASCII (American Standard Code for Information Interchange), dimana tiap karakter diwakili oleh kode 7 bit, dan EBDIC (Extended Binary Code Decimal Interchange Code), dimana 8 bit dipakai untuk mewakili satu karakter.

II.3.1. UNIT INPUT

Komputer menerima informasi yang dikodekan oleh unit-unit output, yang tersusun dari peralatan-peralatan yang dapat membaca data. Contoh yang paling sederhana adalah keyboard. Ia secara elektrik dihubungkan untuk bagian pemrosesan dari pada sebuah komputer.

II.3.2. UNIT MEMORY

Fungsi satu-satunya dari pada unit memory adalah menyimpan program-program dan data. Lagi pula fungsi ini dilakukan dengan bermacam-macam peralatan . Sangat berguna untuk membedakan antara dua kelas dari memory yang termasuk penyimpan primer dan sekunder.

Penyimpan primer, atau memory utama (Main Memory) adalah memory yang cepat diakses dalam pengoperasian elektronic, dimana program-program dan data disimpan selama eksekusi mereka. Memory utama mengandung sel-sel penyimpan semikonduktor dalam jumlah banyak, masing-masingmampu menyimpan satu bit informasi. Sel-sel ini jarang dibaca atau ditulis sebagai sel-sel individual, akan tetapi mereka diproses dalam grup-grup dalam ukuran tetap yang disebut "Word-word". Memory utama diorganisasi sehingga isi dari sebuah word mengandung n bit, dapat disimpan atau diambil dalam satu operasi dasar.

Untuk mengenal secara mudah pada beberapa word pada main memory, sangat berguna untuk membuat nama yang berbeda pada tiap-tiap lokasi word. Nama-nama itu adalah bilangan-bilangan yang mengindentifikasi lokasi-lokasi yang sesuai. Bilangan-bilangan itu dinamakan address. Sebuah word yang disajikan diakses oleh address yang sesuai dan dibutuhkan perintah kontrol untuk menyimpan atau mengambil.

Jumlah bit-bit dalam tiap-tiap word sering dibuat patokan sebagai lebar word dari pada komputer yang diberi-

kan. Komputer-komputer besar biasanya punya 32 atau lebih bit dalam sebuah word, sedangkan mikro komputer dan mini komputer mempunyai lebar word antara 8 sampai 32 bit.. Kapasitas main memory adalah salah satu dari beberapa faktor yang dipakai untuk membedakan karakteristik ukuran dari pada komputer. Mesin-mesin kecil mungkin hanya beberapa ribu word, sedang mesin-mesin besar sering punya berjuta-juta word. Data biasanya dimanipulasi di dalam mesin dalam unit-unit word, penggandaan word atau sub penggandaan word. Akses tipikal ke main memory menghasilkan satu word data yang dibaca atau ditulis ke memory.

Seperti yang disebutkan di atas, program-program dan data harus menetap di main memory selama eksekusi. Instruksi dan data dapat dituliskan di bawah kontrol dari pada unit pemroses. Ia adalah assensial untuk dapat mengakses beberapa lokasi word dalam memory secepat mungkin. Memory dimana beberapa lokasi bisa dilewati oleh address yang sesuai disebut Random Akses memory (RAM) waktu yang dibutuhkan untuk mengakses satu word dinamakan memory access time. Ia adalah waktu tetap, biasanya 100-500 namosikon untuk kebanyakan komputer modern. Penyimpan primer sangat essensial dan cenderung mahal, sehingga penambahan memory sekunder diperlukan untuk menyimpan data yang besar, kadang-kadang sebagian data tak perlu sering diakses. Maka dapat dipilih untuk menyimpan pada disk-disk magnetik, drum-drum dan tape-tape.

II.3.3. UNIT ARITMATIK DAN LOGIKA

Eksekusi dari banyak operasi dalam sebuah komputer diletakkan dalam ALU. Sebagai contoh dua bilangan yang ada pada main memory ditambahkan. Bilangan-bilangan itu dibawa ke ALU setelah diproses oleh penambah hasilnya kemudian disimpan di memory.

Secara sama, operasi aritmatik dan logika yang lain (perkalian, pembagian atau perbandingan bilangan) dikerjakan dengan membawa operand-operand yang dibutuhkan ke ALU, dimana kebutuhan operasi dibentuk. Tidak semua hasil operasi disimpan di main memory, karena prosesor mempunyai sejumlah elemen penyimpan berkecepatan tinggi yang dinamakan register, yang dapat digunakan untuk penyimpan sementara dari pada operand-operand yang sering digunakan. Tiap - tiap register dapat menyimpan satu word waktu akses ke register adalah secara tipical 5 - 10 kali lebih cepat dari pada waktu akses memory.

Unit-unit kontrol dan arimatik biasanya lebih cepat beberapa kali dalam waktu "cycle" dasar dari pada peralatan-peralatan lain yang dihubungkan ke system komputer.

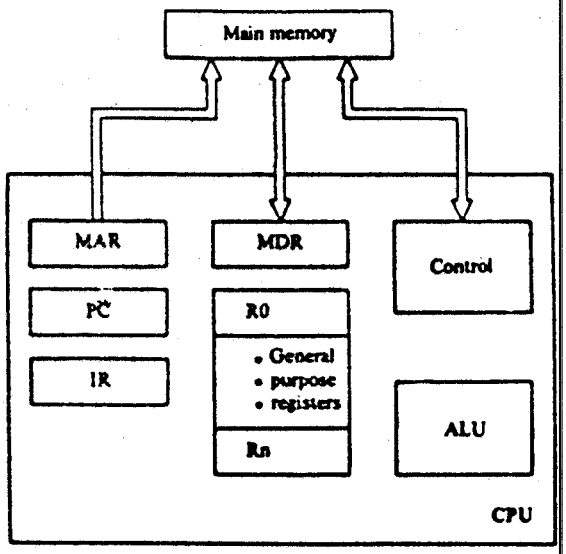
II.3.4. UNIT OUTPUT

Output Unit adalah lawan dari input unit, fungsinya adalah mengembalikan hasil-hasil proces ke bagian luar. Contoh dari Unit Output adalah printer.

II.3.5. UNIT KONTROL

Di atas telah dibicarakan unit-unit untuk penyimpa-

nan dan pemrosesan informasi. Semua itu harus dikoordinasi dengan sesuatu organisasi yang tugasnya dilakukan oleh unit kontrol. Seperti syarat pusat, digunakan untuk mengirim sinyal-sinyal kontrol ke unit-unit lain. Bisa dikatakan secara umum bahwa transfer I/O dikontrol oleh instruksi software.



GAMBAR 2.2

HUBUNGAN ANTARA CPU DAN MAIN MEMORY

II.4. STRUKTUR BUS[3)]

Lebih jauh kita telah mendiskusikan karakteristik-karakteristik fungsional dari masing-masing bagian yang menyusun komputer. Untuk membentuk sistem operasional semua itu harus dihubungkan dalam satu organisasi. Ada beberapa cara untuk melakukan itu, akan dibahas tiga

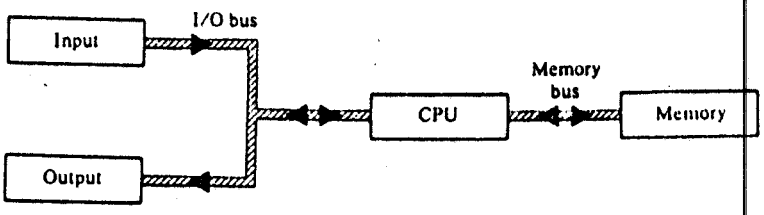
---

3)Ibid hal.11

struktur yang umum.

II.4.1. STRUKTUR DUA BUS

Jika sebuah komputer dibutuhkan untuk operasi kecepatan tinggi, ia harus diorganisasi dengan cara parallel. Yang berarti bahwa semua unit dapat meng"handle" satu data word penuh dalam satu waktu yang juga berarti transfer data antara unit-unit dikerjakan secara parallel, yang berimplikasi bahwa sejumlah banyak kabel/jalur dibutuhkan untuk membuat hubungan yang dibutuhkan. Kumpulan kabel-kabel yang punya identitas gabungan dinamakan bus. Gambar 2.3 memperlihatkan bentuk paling sederhana dari pada struktur komputer dua bus. Processor berinteraksi dengan memory lewat bus memory. Input dan Output lewat bus I / O.



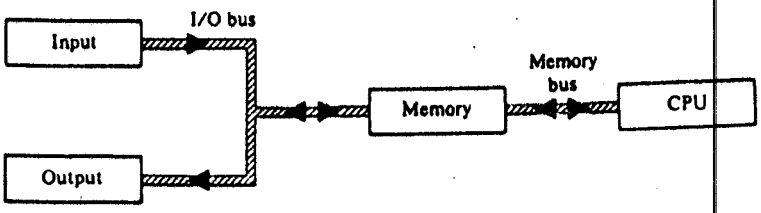
GAMBAR 2.3

STRUKTUR DUA BUS

II.4.2. STRUKTUR DUA BUS ALTERNATIP

Cara yang berbeda dari pada struktur dua bus diberikan pada gambar 2.4. Posisi relatip dari pada prosesor dan memory dibalik, sehingga bus memory berada untuk komunikasi antara keduanya. Akan tetapi, transfer I/O dibuat

secara langsung ke atau dari memory.

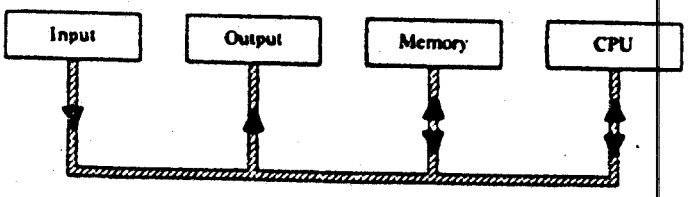


GAMBAR 2.4

STRUKTUR DUA BUS ALTERNATIP

II.4.3. STRUKTUR BUS TUNGGAL

Struktur bus yang lain adalah bus tunggal, semua unit terhubung dalam satu bus, gambar 1-4 menunjukkan struktur bus tunggal.



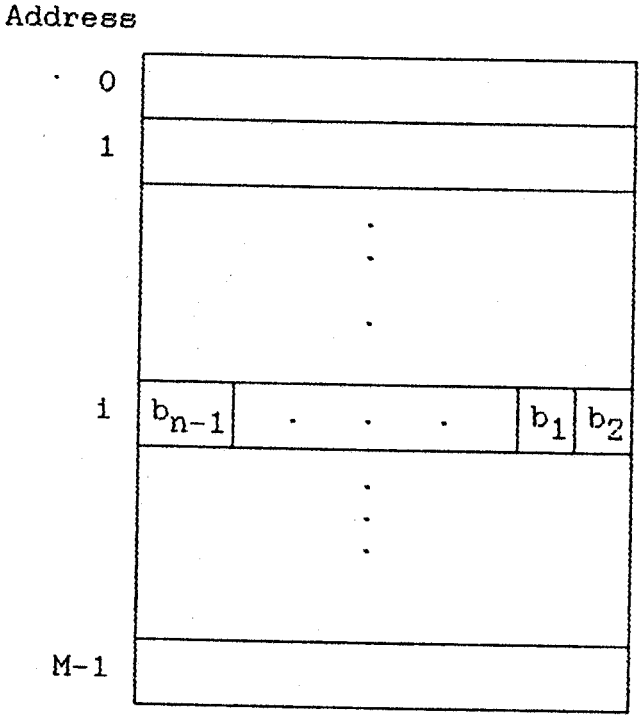
GAMBAR 2.5

STRUKTUR BUS TUNGGAL

II.5. METODE PENGADRESAN

Main memory tersusun dari jumlah yang besar, biasanya ribuan dari pada sel-sel penyimpan. Masing-masing dapat menyimpan satu digit biner atau bit yang mempunyai

nilai 0 atau 2, sehingga 1 bit menyajikan hanya sebuah informasi yang kecil. Bit-bit jarang di"handle" secara individu. Biasanya dalam keadaan grup ukuran yang tepat. Main memory mempunyai n grup bit yang dapat disimpan atau diambil darinya dalam satu operasi dasar. Tiap-tiap grup n bit dinamakan word informasi dan n dinamakan lebar word. Lebar word dalam mikro komputer dan mini komputer berkisar dari 8 sampai 32 bit, sedang komputer-komputer besar biasanya mempunyai 32 atau lebih bit dalam satu word.



GAMBAR 2.6[4)]

ADDRESS-ADDRESS MAIN MEMORY

Untuk mengakses memory untuk menyimpan atau mengambil satu word informasi dibutuhkan nama address yang berbeda tiap-tiap lokasi word. Biasanya memakai angka dari

---

4)Ibid hal.16

0 sampai M-1 address untuk mengalamati lokasi dalam memory yang mengandung M word.

Isi lokasi memory dapat berupa instruksi atau operand, mungkin angka-angka atau karakter-karakter. Untuk address sebanyak k jalur maka lokasi yang dapat dialamati adalah 2^k lokasi.

II.6. METODE PENGADRESAN PADA Z80.[5)]

Prosesor Z80 menyediakan banyak macam instruksi. Operasi data antara register dalam, operasi pada memory luar RAM dan ROM dan operasi terhadap port-port input dan output. Z80 mempunyai register 8 bit sebanyak 18 buah, 16 bit empat buah. Z80 menggunakan address bus 16 bit, sehingga dapat mengakses RAM dan/atau ROM sebanyak 2^{16} atau 65.536 lokasi memory. (biasanya dikatakan 64k), dan dapat memilih sampai 256 kemungkinan I/O. Word Z80 terdiri dari 8 bit data atau biasa disebut byte.

Satu dari keuntungan dari pada Z80 adalah banyaknya metode peng*'address'*an, metode-metode itu antara lain:

- Immediate addressing (pengalamatan langsung)
- Immediate extended addressing.
- Modified page zero addressing.
- Relative addressing.
- Indexed addressing.
- Register addressing.
- Implied register addressing.

---

5) Carr, Joseph J., Z80 Users Manual, Printice-Hall Company, 1980, hal.37

- Register indirect addressing.
- Bit addressing.

Berikut akan dibahas secara ringkas satu persatu.

II.6.1. IMMEDIATE ADDRESSING.

Dalam metode immediate addressing, operand mengikuti opcode pada lokasi urutan berikutnya, dan operand diambil dari lokasi seleksi secara langsung. Contoh utama dari mode immediate addressing adalah Instruksi ADD A,n dan sub A,n. Pada instruksi ini operand n ditambahkan (atau dikurangkan jika Sub A,n) ke isi dari pada akumulator, dan hasilnya disimpan di Akumulator. Format type immediate addressing ditunjukkan berikut :

byte 1 op-code

byte 2 (n).

Misal :

Opcode untuk Instruksi ADD A,n adalah 11 000 110 dalam biner atau C6 dalam hexadecimal. Katakanlah akumulator mengandung A7 (hex) sebelum kode berikutnya dijalankan.

Lokasi memory	code	
0600	C6	byte 1
0601	07	byte 2

Ini berarti instruksi mengambil (C6) adalah instruksi ADD A,n dan operand n (lokasi memory urutan berikutnya) adalah 07 (Hex) sesudah instruksi 2 byte maka:

A ← A + n

A ← A7 + 07

A berisi AE

Kegunaan utama dari pada immediate addressing adalah untuk mengambil data dari register-register spesifik atau lokasi-lokasi atau untuk membentuk operasi-operasi aritmatik menggunakan konstanta-konstanta.

II.6.2. IMMEDIATE EXTENDED ADDRESSING

Adalah Immediate addressing yang dikembangkan sehingga 16 bit data ditransfer. Immediate addressing membutuhkan 3 byte data (opcode dan dua n byte pengikut).

Formatnya adalah :

byte 1 op-code
byte 2 (n1)
byte 3 (n3)

Contohnya adalah instruksi load HL, menyebabkan dua operand di load ke register 16 bit HL. LD HL,nn akan kelihatan:

byte	1	21	op code untuk HL,nn
byte	2	6F	byte orde rendah dari pada HL
byte	3	3D	byte orde tinggi dari pada HL

21 BD FF

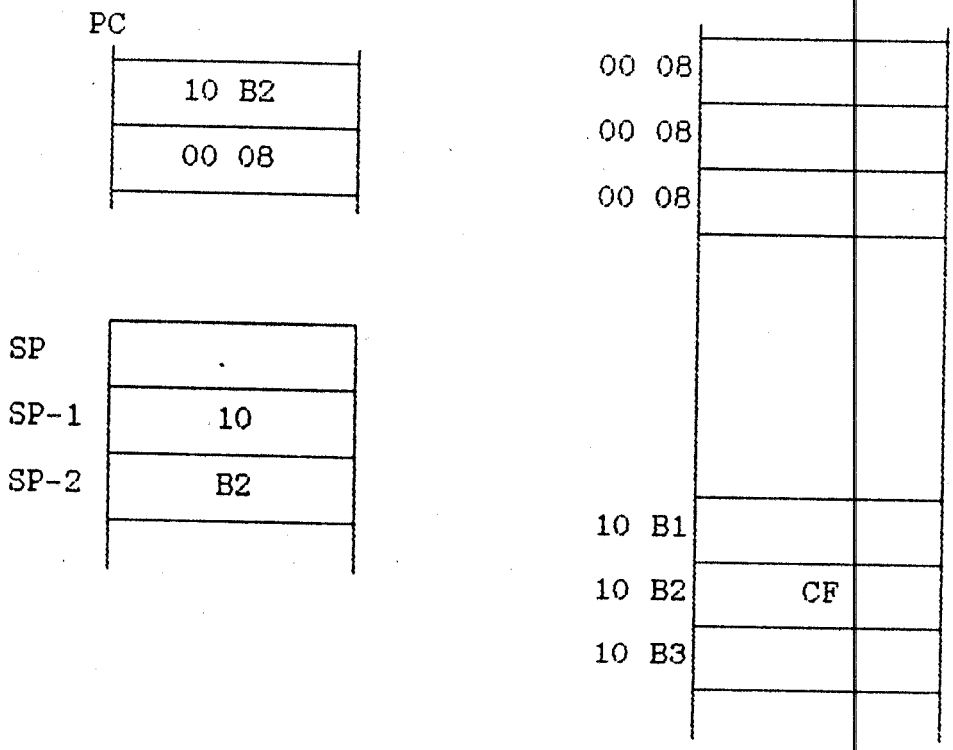
Isi register HL setelah operasi adalah 3D 6F hexa.

II.6.3. MODIFIED PAGE ZERO ADDRESSING

Type instruksi ini mengijinkan programmer untuk mengambil delapan lokasi memory pada halaman nol (yaitu 256 address pertama dari mulai 00 hex). Contoh dari type addressing ini adalah instruksi RST P, bergantung operand

p, akan mereset program counter pada address-address pada halaman nol dari pada memory : 0000, 0008, 0010, 0018, 0020, 0028 dan 0038. Pada instruksi ini isi program counter saat ini di'*push*' ke stack memory external. Byte orde tinggi daripada program counter (PC) di'*load*' dengan 00 (hex). Sedang orde byte rendah di'*load*' dengan satu word yang menyeleksi dari delapan lokasi yang disebut berikut. Sebagai contoh, pengambilan byte orde rendah daripada PC dengan CF (hex) akan menyebabkan instruksi RST 08.

Guna utama dari pada modified page zero addressing addressing adalah membolehkan pelayanan dari pada subroutine-subroutine dengan instruksi "call" byte tunggal. Lihat gambar 2.7.



GAMBAR 2.7

PROSES RESET 08

Pada instruksi di atas akan di eksekusi sebuah program pada halaman 10, dan lokasi 10 B2 di counter, sebuah CF selama satu cycle instruksi fetch. Ini menunjukkan Z80 yang dimodifikasi page zero memanggil sebuah subroutine pada lokasi 00 08, sehingga program kontrol loncat ke lokasi 0008. PC mengandung 10 B2 ketika RST 08 setelah dijalankan. Byte orde tinggi dari pada PC di'*push*' ke lokasi stack pointer (penunjuk stack) SP-1, sementara orde byte tinggi pada lokasi SP-2. Ketika program control kembali (return) dari subroutine, maka PC mengambil kembali data dari stack..

II.6.3. RELATIVE ADDRESSING

Mode relative addressing membolehkan instruksi-instruksi jump dalam 2 byte yang menyebabkan program counter bergeser sebesar integer displacement e . Jump akan diijinkan ke sebuah lokasi seberang -126 sampai +129 dari address sekarang. Instruksi ini adalah instruksi dua byte. Jadi nilai e adalah -128 sampai +127. Integer displacement e selalu menyajikan seperti angka 2'S complement yang ditandai. sehingga nilai-nilai dalam biner akan 1000000 sampai 0111111 (80 to 75 in hex). Karena ini adalah instruksi dua byte, maka jump belum bisa terlaksana sehingga instruksi terselesaikan, program counter akan bertambah dua kali sebelum jump dilaksanakan.

Format instruksi :

byte 1 op code

byte 2 displacement integer (-128 s/d +127)

Gambar 2.8 menunjukkan contoh instruksi jump relatip tanpa kondisi (jr c). Instruksi ini akan menyebabkan bercabang ke subroutine yang dilokasikan oleh displacement e daripada op-code. Nilai dari byte kedua akan e-2. Catatan bahwa op-code tersebut adalah pertama pada lokasi 000A. Setelah eksekusi daripada instruksi ini, PC akan mengandung address baru, 0004 (hex).

Lokasi	Data	
0000		
0001		
0002		
0003		
0004		PC sesudah dieksekusi
0005		
0006		
0007		
0008		
0009	18	op-code untuk JR,R
000A	FA	2's complement untuk -6
000B		
(dst)		

GAMBAR 2.8

INSTRUKSI JUMP RELATIVE

II.6.4. EXTENDED ADDRESSING

Pada extended addressing, diijinkan untuk menggunakan 2 byte memory untuk membentuk address 16 bit. Pada type extended addressing, akan terjadi satu atau dua op-code, yang diikuti oleh dua byte address atau operand. Dalam kasus ini byte pertama adalah byte orde rendah, sedang

yang ke dua adalah orde tinggi .

Format :

byte 1 op-code

byte 2 (mungkin op-code tambahan)

byte 3 n1

byte 4 n2

Misal :

LD A, (nn) adalah sebuah instruksi extended addressing yang berarti mengisi akumulator dengan byte yang lokasinya ditunjukkan oleh operand nn. Atau lebih jelas , lokasi tersebut berisi data yang akan diisikan ke akumulator.

```
byte   1    3A   ;Op-code untuk  LD A,(nn).
byte   2    n1   ;low
byte   3    n2   ;high
```

II.6.5. INDEXED ADDERSSING

Type ini adalah peng*'address'*an menggunakan dua register index 16 bit (IX dan IY), plus sebuah nilai displacement penyerta op-code, untuk menghitung address efectif dari pada sebuah jump pada type instruksi indexed addressing, dimana akan ada dua byte opcode, diikuti dengan integer displacement d.

Format:

byte 1 (op - code)

byte 2 (op - code)

byte 3 (d)

Seperti instruksi LD (IX + d),A. Instruksi ini menyebab-

kan lokasi memory ditunjukkan oleh isi dari register index IX dan displacement integer d yang diisi dari akumulator.

Kode berikut :

byte 1 DD { LD (IX + d),A }

byte 2 77 { LD (IX + d), A }

byte 3 d displacement.

Akumulator mengandung 3F (hex), dan index register IX mengandung 4400 (hex). Kode DD 7705 dijalankan pada instruksi fetch. Ini mengindikasikan instruksi LD (IX + D),A. Dimana d adalah 05 (hex). Lokasi memory 4400 + 05 = 4405 akan mengandung 3F (hex) setelah eksekusi instruksi ini.

II.6.6. REGISTER ADDRESSING

Mode addressing ini bisa untuk mentransfer data antara register Z80 seperti instruksi LD R, R'. Register A, B, C, D, E, H dan L dapat dipakai sebagai R maupun R'. Hanya perlu 1 byte op-code yang dipakai untuk menjalankan instruksi ini.

Format instruksi ini:

01 | - r - | - r' - |

Misal:

Kode register untuk register D adalah 010, dan kode register E adalah 011. Sedang operasi dari pada instruksi LD r,r' berarti r ← r' (register r disalinkan dari register r'). Jika diinginkan menyalin register E ke register D maka op-code

adalah 01 010 011 dalam hexadeciamal 53.

II.6.7. IMPLIED ADDRESSING

Pada mode implied addressing dipakai instruksi spesial yang selalu menggunakan register CPU yang sama untuk mengisi operand-operand. Contoh type instruksi ini adalah LD R,A yang akan berarti menyalin ke register R (Refresh memory) dengan isi dari akumulator.

II.6.8. REGISTER INDIRECT ADDRESSING

Instruksi ini akan menyebabkan transfer data antar CPU dan lokasi memory yang ditunjukkan oleh isi dari salah satu pasangan register - register 16 bit. Contoh dari type instruksi ini adalah LD (DE),A. Mnemonic ini adalah dibaca *'Salin data ke lokasi memory yang ditunjukkan oleh register pasangan DE dengan isi dari pada akumulator.*

Misal:

Akumulator berisi 9D (hex) dan register pasangan DE berisi 65 08 (hex). Jika perintah di bawah ini dijalankan: 00 010 010 (op-code untuk LD (DE),A). Maka lokasi memory 6508 akan berisi byte 9D (hex) setelah eksekusi instruksi tersebut.

II.6.7. BIT ADDRESSING

Satu dari kelebihan yang prinsip dari Z80 dalam banyak type dari pada pemrograman adalah kemampuan untuk mengeset, mereset atau test kondisi dari single bit dari beberapa register. Op-code yang akan dibentuk:

byte 1 (op-code)

byte 2 01 b r

Misal :

Kode bit (b) untuk bit 5 adalah 101, dan kode register D adalah 010. Pada orde test kondisi dari pada bit 5 dalam register D dipakai instruksi bit 5,D dengan op-code:

byte 1 11 001 011

byte 2 01 101 010

Dalam kasus ini, flag Z dari pada register F (flag) akan SET jika bit yang ditest adalah 0 dan RESET jika bitnya 1.

II.7. INSTRUKSI - INSTRUKSI Z80 SECARA UMUM

Ada 128 instruksi yang dapat dieksekusi oleh Z80 secara aktual, jika dijabarkan dapat menjadi 400 lebih instruksi. Yang lalu telah diklasifikasikan pada pembahasan mode addressing. Pada bagian ini akan diklasifikasikan instruksi-instruksi dalam grup, masing-masing grup itu adalah:

- Load dan Exchange
- Transfer blok dan Search
- Arithmetic and Logical Instruction
- Rotate and Shift
- Bit manipulation
- Input/output

II.7.1 INSTRUKSI - INSTRUKSI LOAD

Instruksi-instruksi load dipakai untuk memindahkan data dari lokasi satu ke yang lokasi lain.
Ada dua type dasar dari perintah load:

a. move data antar register internal
b. move data dari/ke register internal ke/dari lokasi memory external.

Ada dua grup dasar dari pada instruksi load yaitu 8 bit dan 16 bit instruksi load. Instruksi-instruksi move byte tunggal pada data, menggunakan 8 bit register dan 8 bit lokasi memory (kadang-kadang ditunjukkan oleh 16 bit register index).

Instruksi 16 bit menggunakan register-register pasangan AF, BC, DE, HL dan SP, IX, IY. Semua selalu menggunakan 2 byte address memory untuk menspesifikasikan lokasi-lokasi memory dari dua byte data yang akan disalin ke CPU atau dari CPU.

Dalam semua perintah LOAD harus dispesifikasikan baik sumber maupun tujuannya.

II.7.2. INSTRUKSI - INSTRUKSI EXCHANGE

Instruksi-instruksi exchange dipakai untuk tukar menukar dari pada isi dua register yang dispesifikasikan. Instruksi-instruksi exchange menggunakan EX dan EXX.

II.7.3. BLOK TRANSFER DAN BLOK SEARCH

Salah satu yang paling menyenangkan dari pada Z80

adalah BLOK TRANSFER dan BLOK SEARCH. Instruksi transfer meliputi : LDI, LDIR, LDD, dan LDDR, sedang instruksi-instruksi blok meliputi: CPI, CPIR, CPD, CPDR. Instruksi-instruksi blok menggunakan 3 register pasangan 16 bit dalam eksekusinya:

HL address lokasi sumber

DE address lokasi tujuan

BC byte penghitung

Dalam banyak program menggunakan instruksi-instruksi tersebut, ini memerlukan register (HL, DE, BC) untuk menginisalisasi nilai yang dibutuhkan. Ketika instruksi blok dijalankan, register - register tersebut naik/turun secara otomatis ke lokasi berikutnya:

Instruksi-instruksi blok adalah sebagai berikut:

LDI:

Load dan increment/naik. instruksi ini menyalin 1 byte data dari lokasi yang ditunjukkan oleh pasangan register DE, setelah instruksi LDI register-register HL dan DE ditambahkan (ke lokasi berikut), dan register BC dikurangkan.

LDIR :

Sama seperti LDI hanya perintah ini berulang hingga register BC bernilai 0. Jika register BC diisi dengan bilangan tertentu, maka instruksi ini dapat dipakai untuk memindahkan data dari blok yang satu ke blok yang lain.

LDD dan LDDR :

LDD dan LDDR mempunyai persamaan dengan LDI dan LDIR

hanya bedanya register - register HL dan DE pada perintah ini dikurangkan tiap selesai eksekusi.

II.7.4. INSTRUKSI BLOK SEARCH

CPI: Compare And Increment.

Instruksi type ini akan membandingkan (compare) isi dari pada akumulator oleh pasangan register HL. Hasil dari pada perbandingan ditandai oleh kondisi bit-bit register flag. Setelah eksekusi dengan instruksi ini pasangan register HL isinya ditambah-kan/dinaikkan, dan isi register pasangan BC sebagai counter.

CPIR:

Sama dengan CPI akan tetapi diulang sampai salah satu kondisi terpenuhi :

a. byte counter (BC) = 0

b. data pada memory address sama dengan data pada Akumulator.

CPD: Compare dan Decrement.

CPDR: Compare, Decrement dan Repeat.

Keduanya sama dengan CPI dan CPIR , hanya HL dalam hal ini dikurangkan.

II.7.5. INSTRUKSI-INSTRUKSI ARITMATIK DAN LOGIKA

Ada dua grup instruksi aritmatik dan logika: 8 bit dan 16 bit. Dalam grup aritmatik 8 bit adalah penambahan (ADD), pengurangan (SUB), pengurangan dengan (SBC). Grup

logika 8 bit : AND, OR dan XOR instruksi. Juga yang termasuk klasifikasi instruksi 8 bit adalah compare (CP), increment (INC) dan decrement (DEC).

Dasar yang sama instruksi ADD, ADC, SBC, INC dan DEC dipakai dalam grup aritmatik/logika 16 bit, akan tetapi berlainan dengan register-register 8 bit (A, B, C, D, E, H dan L), lokasi-lokasi memory tunggal, grup 16 bit memakai pasangan-pasangan register (HL, IX, IY, BC, DE dan SP).

Semua selalu diberi penyerta data instruksi-instruksi Desimal Adjust the accumulator (DAA) . Yang bisa dipakai untuk multipresesi angka angka BCD, bilangan-bilangan bertanda atau tidak bertanda, atau 2'S complement angka-angka bertanda.

Ada 4 instruksi tambahan dalam grup aritmatik/logika yakni : Complement Akumulator (CPL), Negasi Akumulator (NEG), Complement Carry Flag (CCF) dan Set Carry Flag (SCF). Instruksi CPL menyebabkan Akumulator dikomplemenkan, yang berarti bahwa 1 menjadi 0 dan 0 menjadi 1. Instruksi NEG menyebabkan isi dari Akumulator di ekspresikan dalam bentuk 2'S complement. Instruksi CCF menyebabkan Carry Flag dikomplemenkan; 0 menjadi 1 dan 1 menjadi 0. instruksi SCF menjadikan CF (Carry Flag) SET atau berlogika 1.

II.7.6. INSTRUKSI ROTATE DAN SHIFT

Grup instruksi rotate dan shift meliputi: RLC, RRC, RL, RR, SLA, SRA, SRL, RLD, RRD, RLCA, RRCA, RLA, dan RRA.

II.7.7. INSTRUKSI-INSTRUKSI MANIPULASI BIT

Salah satu yang membuat Z80 lebih baik dengan μP yang lain adalah adanya instruksi manipulasi bit. Fungsinya dapat men 'test' sebuah bit apakah 1 atau 0, juga dapat mereset sebuah bit (RES) atau menset sebuah bit (SET). Bagian bit yang dites, set atau reset dapat dibentuk pada register (A, B, C, D, E, H dan L) atau lokasi memory yang lain. Dalam kasus yang terakhir, digunakan register index atau register indirect addressing untuk menyeleksi lokasi memory. Disini terdapat banyak jumlah instruksi-instruksi individu dalam grup ini, karena dapat dipilih sampai 8 bit (0 - 7) dari tujuh register yang berbeda atau memory yang ditentukan oleh register-register HL, IX dan IY.

II.7.8. INSTRUKSI CALL, JUMP DAN RETURN

Sebuah komputer digital menjalankan atau mengeksekusi instruksi-instruksi dengan cara urut. Dalam keadaan biasa program counter ditambah satu untuk menghitung dalam tiap-tiap instruksi yang dijalankan. Jumlah angka yang dihitung naik ditentukan oleh jumlah byte yang dibutuhkan untuk bagian type instruksi. Walaupun urutan penjalanan ini adalah aspek penuh dari pada komputer digital, ia akan selalu limit range dari pada problem-problem yang mungkin yang dapat diselesaikan pada yang suka menurut bentuk pemrosesan secara urut. Ia akan tak mungkin untuk memben-

tuk banyak operasi yang membutuhkan seperti keputusan yang paling sederhana. Seperti hal sederhana dari pada pemasukan data akan menjadi tidak mungkin. Pada operasi sebuah port input yang dihubungkan ke keyboard. Kita membuat sebuah instruksi *jump* jika test-test strobe bit (biasanya bit 7), dan jika tak di dapat, jump-jump kembali ke awal dari pada loop. Jika didapat maka sebaliknya, program dapat diizinkan lewat pada instruksi berikutnya (biasanya instruksi input) ini hanya contoh sederhana. Banyak problem yang membutuhkan keputusan logika pada bagian dari pada komputer, tak dapat dibentuk tanpa memakai instruksi-instruksi *jump*, *call* dan *return*. Instruksi-instruksi itu dalam grup meliputi: *jp*, *jr*, *call*, *djnz*, *ret dan retn*.

Sebuah instruksi jump (JP) adalah sebuah cabang ke sebuah subroutine pada address lain kemudian address berikut dalam urutannya. Address pada instruksi berikut (yaitu instruksi subrotine) adalah diisikan ke program counter (PC). Dapat dilakukan dengan tiga mode addressing : Immediate addressing, register indirect dan relative.

Tiap-tiap perbedaan tipe dari pada instruksi *jump* dikunci untuk kondisi-kondisi tertentu yang direfleksikan oleh status bit-bit dari pada register flag (F). Kondisi yang kemungkinannya dapat ditentukan oleh pemilihan dengan op-code yang meliputi carry, non carry, zero, non zero, parity even, parity odd, tanda negatif, tanda positif dan tanda kondisi.

Kondisi instruksi-instruksi jump mencari status bit yang disediakan register flag. Jika kondisi ketemu, maka

kondisi jump disetujui.

Pada immediate extended addressing, jump membolehkan pada address memory 16 bit yang ditentukan oleh dua bit pengikut instruksi jump. Jika kondisi panggil ketemu, maka program kontrol akan digeser ke lokasi yang ditentukan oleh dua bit pengikutnya.

Register indirect addressing dibolehkan untuk menyimpan address 16 bit dari pada instruksi pertama dalam subroutine yang dikehendaki untuk diekseekusi dalam salah satu dari tiga pasang register. Register-register HL, IX atau IY dapat ditentukan dengan penyediaan dari pada op-code, instruksi-instruksi jump register inderect semua tanpa kondisi.

Instruksi jump relative addressing (JR) mengandung kondisi-kondisi sebagai berikut: carry, non carry, zero, non zero dan tanpa kondisi. Pada tiap instruksi jump ini, instruksi berikutnya dari program yang akan dijalankan (yaitu op-code instruksi pertama pada subroutine) ditentukan dengan isi PC sekarang ditambah dengan displacement e. Nilai e dapat berkisar -126 s/d +129.

Dalam kondisi salah satu dari instruksi jump tak berkondisi, program counter diisi secara langsung dengan dua byte pengikut op-code untuk jump. Byte ke-2 dari pada 3 byte instruksi menjadi byte orde rendah dari pada address PC, sedang byte ke-3 dari pada instruksi menjadi orde tinggi dari pada address PC. Karena isi PC sekarang berubah, maka program kontrol ditransfer ke lokasi instruksi yang ditentukan oleh PC.

Ada bentuk instruksi jump spesial yang sangat berguna yaitu DJNZ, ia akan mengurangi register B dan jump jika non zero. Instruksi ini mengizinkan kita untuk menggunakan register B sebagai byte counter. Register B diisi dengan integer yang sesuai dengan hitungan untuk menjalankan sebuah subroutine. Program kontrol akan mentransfer ke subroutine yang ditentukan oleh integer displacement e, (ini relative addressing) setelah DJNZ dijalankan. Ketika program kontrol kembali (return), register B dikurangkan. Jika operasi tak membawa register B ke arah 0 (nol), maka jump dijalankan lagi. Ia akan terus sehingga B menjadi 0. Jika B sebelumnya diisi 0 maka program akan jump dan loop sampai 256 penuh dan berhenti hingga B nol lagi.

Instruksi CALL adalah kasus spesial dalam instruksi JUMP. Jika CALL digunakan, maka address pada lokasi memory mengikuti instruksi CALL secara langsung diisikan ke stack memory external (ditunjukkan oleh isi register SP) ini membolehkan untuk bercabang ke subroutine dan kemudian kembali ke urutan program utama.

Instruksi return (RET) adalah kebalikan instruksi CALL, jika digunakan untuk kembali ke main program dari subroutine oleh instruksi CALL setelah berakhir. Instruksi RET biasanya akhir dari pada subroutine. ketika instruksi REt dijalankan PC diisi dengan stack memory external (ditunjukkan oleh SP). Ini akan menjadi address daripada instruksi pertama pengikut instruksi CALL yang menyebabkan cabang subroutine.

Ada dua instruksi return khusus , RETI dan RETN, ini adalah untuk kembali ke program kontrol utama setelah pelayanan sebuah respon interrupt atau non maskable interrupt.

II.7.9. INSTRUKSI INPUT / UOTPUT

Instruksi ini menyebabkan data diinputkan ke, atau dioutputkan dari CPU. Z80 mempunyai banyak register berbeda, dan punya instruksi-instruksi I/O yang dapat menggunakan register-register tersebut tanpa memerlukan dulu seorang programmer untuk menstransfer isi dari pada akumulator, pertambahan ada beberapa instruksi blok I/O.

Immediate addressing dapat menggunakan akumulator (reg. A), Sementara register - register B, C, D, E, H dan L digunakan untuk register indirect addressing. Instruksi blok I/O selalu menggunakan register indirect.

Mungkin banyak instruksi I/O adalah input langsung dan output langsung [IN A,(n) dan Out (n),A]. Pada instruksi-instruksi ini operand n adalah address 8 bit salah satu dari pada 256 port kemungkinan (000 - 255). Address ini akan disediakan pada 8 bit rendah (A0 - A7) dari pada address bus. Data input dan output dilewatkan pada 8 bit data bus ke atau dari akumulator. Kasus dari pada register indirect addressing, isi dari pada register C menentukan 8 bit address port yang diseleksi. Address ini (isi daripada reg. C) dilewatkan pada byte address rendah ke sinyal peralatan yang di pilih.

Instruksi blok input meliputi INI, INIR, IND dan INDR, Instruksi blok output meliputi : OUTI, OUTIR, OUTD dan UOTDR. Analogi dengan instruksi transfer blok memory, kecuali menggunakan pasangan register HL untuk menunjukkan 8 bit address I/O , pada lokasi memory external. Pada instruksi ini register B digunakan sebagai byte counter, seperti kasus register indirect diatas, register C dipakai sebagai address port I/O.

II.7.10. INSTRUKSI-INSTRUKSI CONTROL CPU

Ada tujuh instruksi dalam Z80 yang digunakan untuk control dari pada CPU, yaitu : NOP, HALT, DI, EI, IM0, IM1 dan IM2.

- Instruksi NOP adalah instruksi yang berarti tak ada operasi selama eksekusi NOP, CPU tak mengerjakan apa-apa secara mutlak.
- Instruksi HALT menyebabkan CPU berhenti beroperasi hingga interrupt diterima, instruksi DI tak membolehkan interrupt, sedangkan EI membolehkan interrupt.
- Instruksi IM0, IM1, dan IM2 membolehkan programmer untuk men'set' sebanyak tiga mode interrupt. Mode nol (IM0) menyebabkan Z80 berpikir seperti 8080A. IM1 menyebabkan program control mentransfer secara otomatis ke lokasi 0038 (hex) ketika interrupt

diterima. Interrupt mode 2 (IM2) membolehkan call tak langsung ke sebuah subroutine interrupt pada lokasi yang ditentukan oleh dua byte : isi register I dan word 8 bit yang diterima dari peralatan interrupt.

II.8. FLAG-FLAG DALAM Z80

Register-register F dan F' dalam chip CPU Z80 digunakan sebagai flag-flag kondisi. Enam dari delapan bit dalam register di'SET' (dibuat sama dengan 1) atau RESET (dibuat sama dengan 0) yang tergantung dari kondisi-kondisi yang dihasilkan dari macam-macam operasi CPU. Isi-isi register flag tergantung pada progammer. Bit-bit dari register flag ditunjukkan pada gambar 2.9.

BIT NO.	DEFINITION
0	C
1	N
2	P/V
3	-
4	H
5	-
6	Z
S	S

GAMBAR 2.9

BIT-BIT DARI REGISTER FLAG

- Carry Flag C mennunjukkan bahwa ada bit carry dari bit orde tertinggi dalam akumulator. Flag akan di'set' jika ada *carry* dari operasi penambahan, atau *borrow* selama operasi pengurangan, atau dalam kasus-kasus tertentu seperti pada operasi-operasi

shift dan rotate.

- Flag Zero Z di'set' (Z=1) jika operasi mengasilkan harga nol (yaitu 00000000_2) yang akan disimpan dalam akumulator, jika hasilnya selain nol maka flag Z adalah reset (Z=0)
- Flag tanda S (sign) menyimpan keadaan bit 7 daripada akumulator (1 atau 0). Bit 7 merupakan tanda bilangan. Jika bilangan negatip maka bit 7 adalah 1, jika bilangan berharga nol atau positip maka bit 7 adalah 0.
- Flag P/V (parity/overflow) mempunyai dua keadaan. Pada operasi-operasi logika (AND, OR,XOR) flag P/V mengindikasikan parity daripada isi akumulator (genap atau ganjil). Pada operasi-operasi 2's complement flag P/V mengindikasikan apakah overflow atau tidak. Kondisi overflow mengidikasikan jawaban pada akumulator adalah error, karena melampaui range bilangan yang diijinkan (-128 s/d +127). Flag P/V di set jika dalam keadaan overflow.

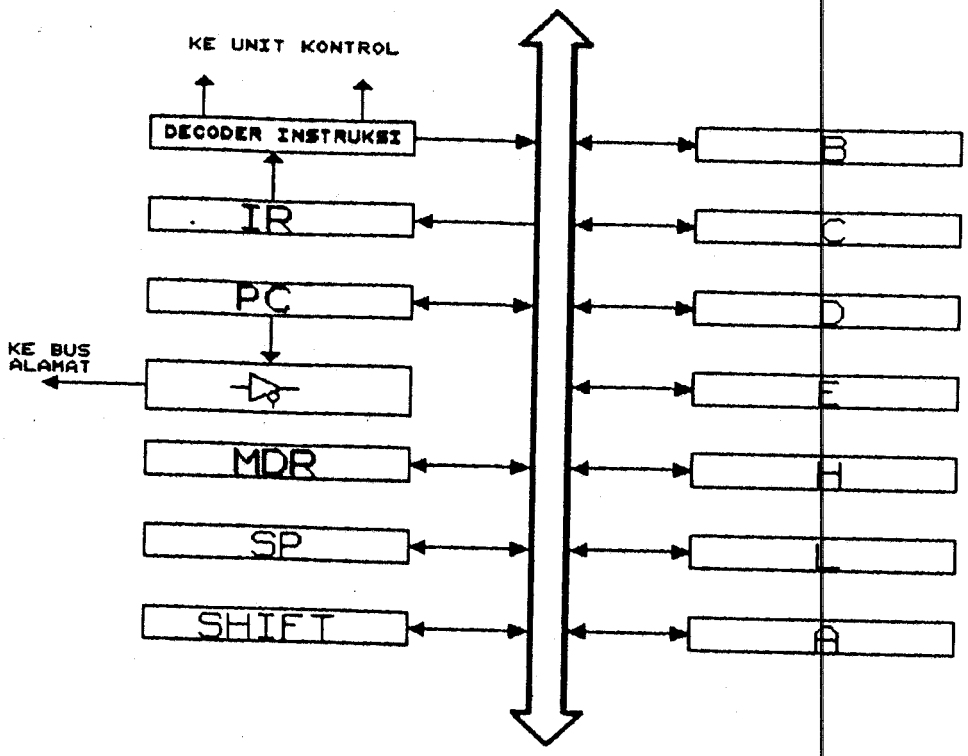
  Pada kasus operasi-operasi logika, flag P/V akan di'set' untuk parity ganjil dan reset untuk parity genap.
- Flag H (half-carry) adalah borrow atau carry sebuah BCD dari setengah byte rendah daripada isi akumulator. secara sama, flag pengurangan (N) digunakan untuk koreksi (penepatan decimal) pada operasi-operasi pengurangan.

BAB III
PERENCANAAN

III.1 PENDAHULUAN

Pada bab ini akan dibahas tentang perencanaan yang berhubungan dengan alat yang akan dibuat yang meliputi rangkaian-rangkaian yang terdapat pada blok diagram. Untuk rangkaian yang sejenis yang ada pada sub-sub bab yang sekiranya dapat dipahami dengan sebagian contohnya akan disajikan pada lampiran.

III.2 BLOK DIAGRAM



GAMBAR 3.1

BLOK DIAGRAM ALAT YANG AKAN DIRENCANAKAN

III.3 CARA KERJA SISTEM

Op-code terlebih dahulu ditangkap dari pembacaan memory luar ke dalam register instruksi (IR) sampai tiga byte, dari tiga byte tersebut kemudian diterjemahkan oleh decoder-decoder ke perintah-perintah yang sesuai, selanjutnya penangkapan op-code atau data disesuaikan dengan jumlah byte yang telah diterjemahkan, yang dalam perencanaan ini meliputi op-code/data dari satu byte hingga tiga byte.

III.4. RANGKAIAN CLOCK INTERNAL

Untuk membaca memory diperlukan waktu tertentu sehingga cukup untuk menyajikan data secara sempurna. Untuk membuat waktu tersebut digunakan rangkaian single shot multivibrator dari IC 74ls221 dimana waktunya dapat diperoleh dengan menambahkan rangkain R dan C pada bagian luarnya. Output dari rangkaian tersebut disamping digunakan sebagai clock internal, juga pada level 'low'nya digunakan sebagai memory request (MREQ) dan sinyal read (RD). Gambar 3.2 menunjukkan bagaimana rangkaian tersebut dibuat.

- MVA direncanakan dengan R = 10 K dan C = 100 pF sehingga diperoleh Tw:

$$Tw = 0,7\ RC$$
$$= 0,7 \times 10 \times 10^{3} \times 100 \times 10^{-12}$$
$$= 0,7 \times 10^{-6}$$

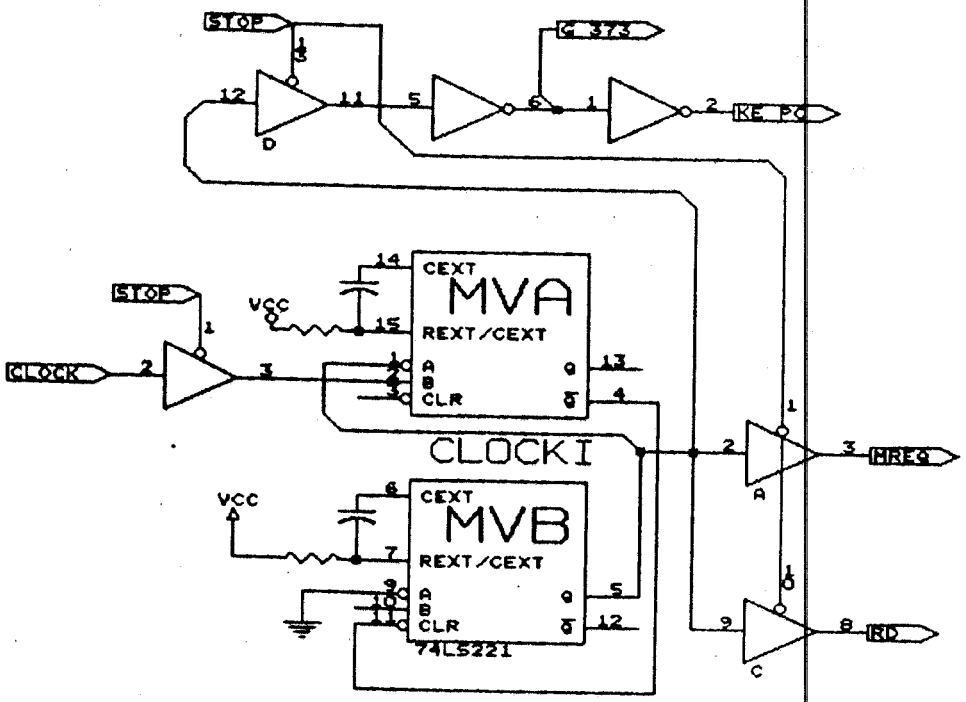
$= 700 \times 10^{-9}$

$= 700$ ns

TABEL 3.1

TABEL KEBENARAN IC 74LS221

INPUTS			OUTPUTS	
CLEAR	A	B	Q	-Q
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	⊓	⊔
H	↓	H	⊓	⊔
↑	L	H	⊓	⊔



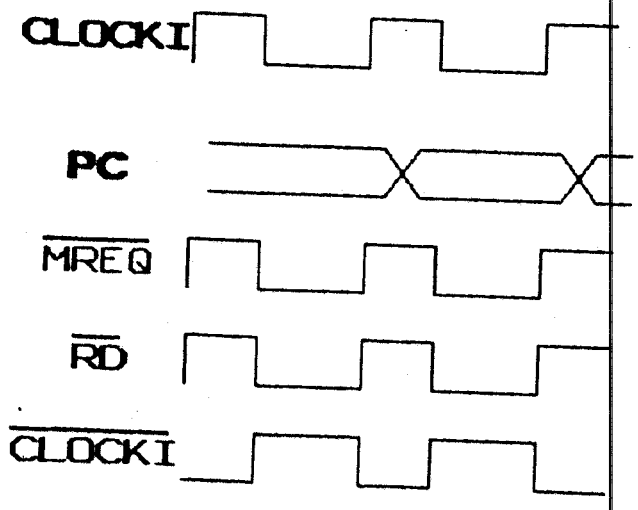
GAMBAR 3.2

RANGKAIAN CLOCK INTERNAL

- MVB direncanakan dengan R = 4,7 K dan C = 110 pF sehingga diperoleh Tw:

$$Tw = 0,7\ RC$$
$$= 0,7 \times 4,7 \times 10^{3} \times 110 \times 10^{-12}$$
$$= 0,7 \times 0,517 \times 10^{-6}$$
$$= 362 \times 10^{-9}$$
$$= 362\ ns$$

Dari hubungaan rangkaian pada gambar diperoleh harga level high sebesar 362 ns, sedang level *low* punya harga tak tentu akan tetapi punya harga minimal sebesar 700 ns. Hal ini bisa terjadi karena selama input A_{MVA} *high* maka MVA dalam keadaan normalnya ($-Q_{MVA}$ *high*), sedang pada saat A_{MVA} *low* maka input B_{MVA} menunggu sinyal yang naik (dari *low* ke *high*). Dari rangkaian dapat diperoleh diagram pewaktu seperti terlihat pada gambar 3.3

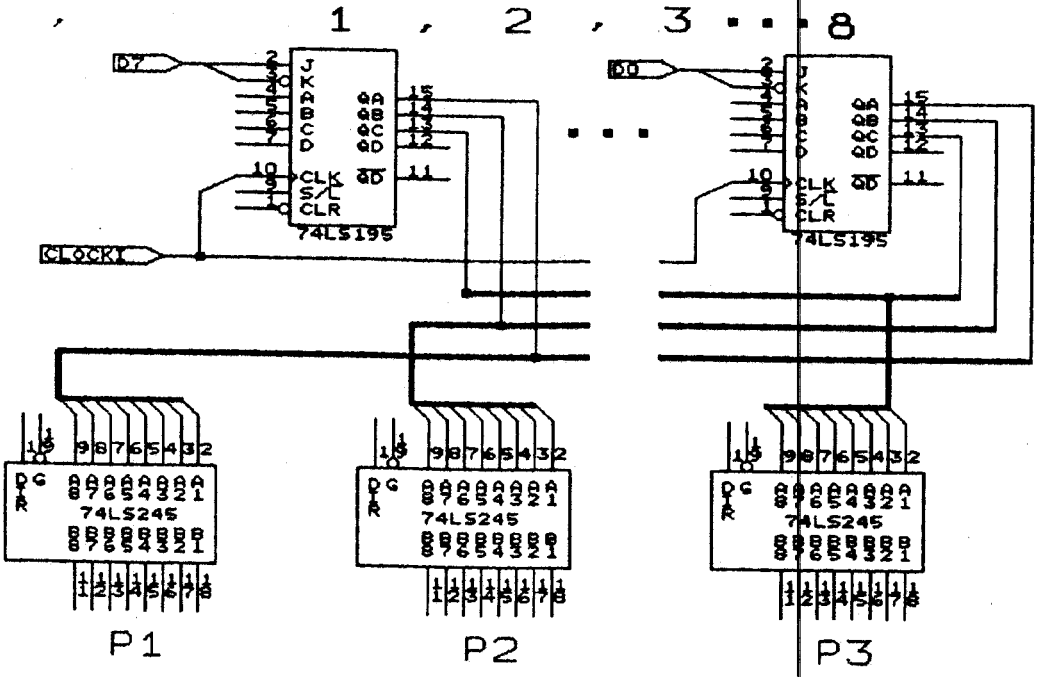


Gambar 3.3

Diagram pewaktu untuk PC, MREQ, dan READ

III.5 REGISTER INSTRUKSI (IR)

Sebelum kode-kode operasi yang ada di memory external diterjemahkan, terlebih dahulu data tersebut ditangkap ke suatu register yang biasa disebut register instruksi.



GAMBAR 3.4

RANGAKAIAN REGISTER INSTRUKSI

Dalam perencaan ini direncanakan tiga buah register instruksi yang selanjutnya ditandai dengan P1,P2 dan P3. Data dari register P3 digeser ke rgister P2 kemudian ke P1, sedang P3 diisi dari memory external. Pada saat reset pengisian .dilakukan tiga kali, sehingga semuanya berisi data baru, sedang untuk selanjutnya tergantung kode operasi sebelumnya dimana kode operasi tersebut membutuhkan

satu hingga tiga byte data. Penerjemahan dilakukan di P1 terlebih dahulu, selanjutnya tergantung kode operasi yang bersangkutan yang mungkin membutuhkan data pada P2 atau sampai P3. Jika hanya satu byte yang dibutuhkan dalam operasi tersebut, maka hanya perlu penggeseran satu kali. Begitu juga untuk operasi-operasi yang membutuhkan dua atau tiga byte, maka setelah operasi-operasi tersebut digeser dua atau tiga kali. Gambar 3.4 memperlihatkan rangkaian register instruksi. Register-register tersebut direncanakan dari delapan buah IC shift register 4 bit 74LS195 dimana 3 bit nya yang dipakai sehingga jika dikalikan dengan delapan akan menjadi tiga buah register P1,P2 dan P3.

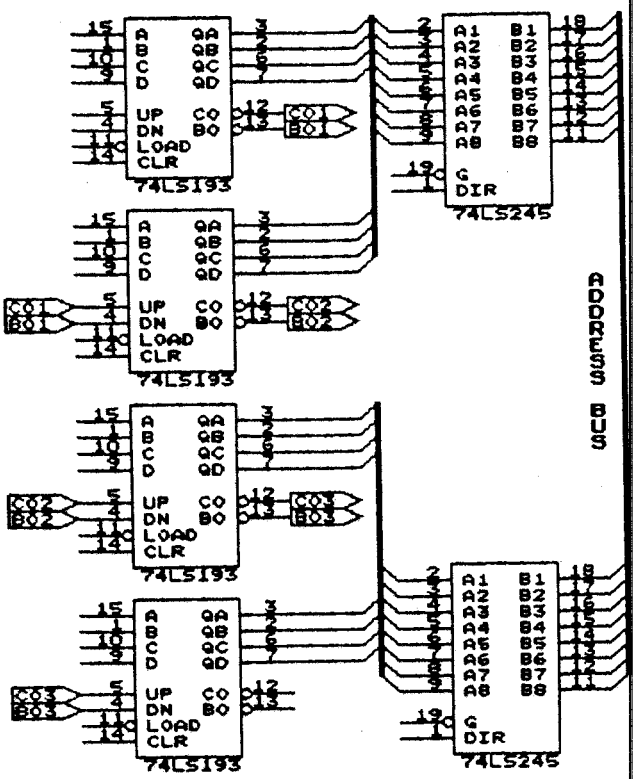
TABEL 3.2

TABEL KEBENARAN IC 74LS195

INPUTS										OUTPUTS			
CLEAR	SHIFT/ LOAD	CLOCK	SERIAL		PARALLEL					QA	QB	QC	QD
			J	K	A	B	C	D					
L	X	X	X	X	X	X	X	X		L	L	L	L
H	L	↑	X	X	a	b	c	d		a	b	c	d
H	H	L	X	X	X	X	X	X		Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
H	H	↑	L	H	X	X	X	X		Q_{A0}	Q_{A0}	Q_{Bn}	Q_{Cn}
H	H	↑	L	L	X	X	X	X		L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	↑	H	H	X	X	X	X		H	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	↑	H	H	X	X	X	X		$\overline{Q_{An}}$	Q_{An}	Q_{Bn}	Q_{Cn}

III.6. PROGRAM COUNTER

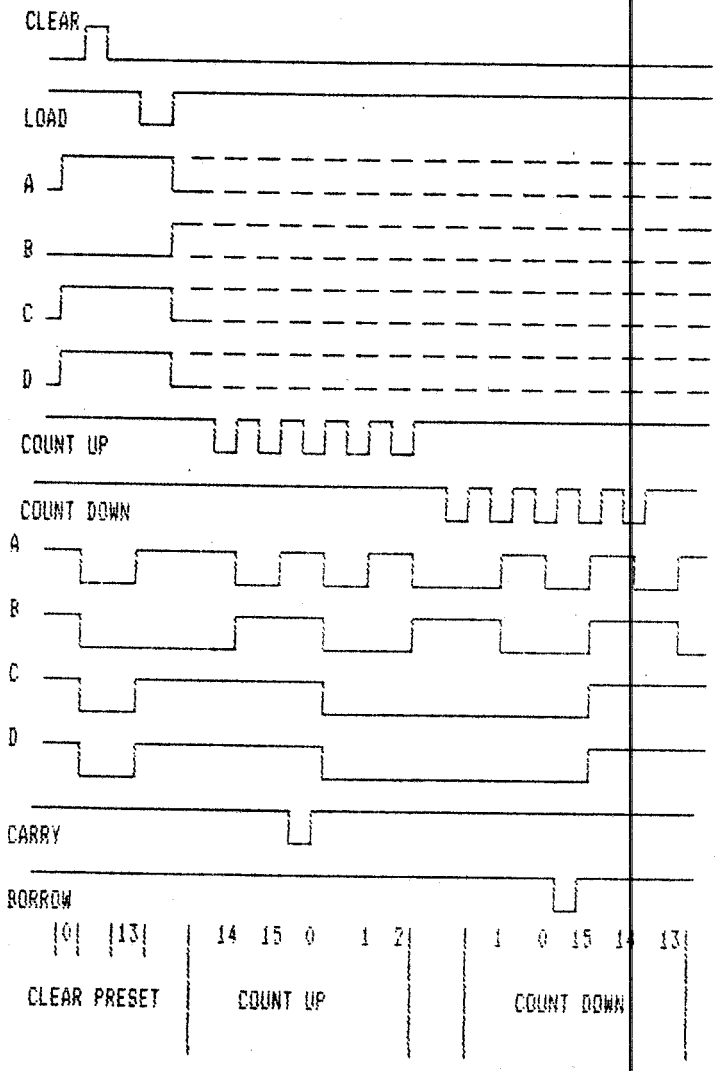
Untuk membuat *processor* agar dapat menjalankan program secara urut, diperlukan counter. Program counter (PC) berfungsi sebagai pengurut program yang akan dijalankan dengan cara mengeluarkan data pada PC tersebut ke jalur address bus. Kecuali untuk kasus-kasus tertentu yaitu perintah Jump, call, Ret, Push dan Pop maka PC diisi dengan data tertentu yang selanjutnya proses berjalan secara urut yang dimulai dari alamat yang baru itu, yaitu data yang baru diisikan. Pada gambar 3.5 terlihat bagaimana rangkaian PC tersebut disusun.



GAMBAR 3.5

RANGKAIAN PROGRAM COUNTER

Hubungan PC dengan data bus berguana untuk menyimpan atau mengambil kembali PC dari Stack yang terjadi pada proses CALL, RETURN, PUSH dan POP. Hubungan dengan data tersebut dipisah jadi dua bagian yaitu byte rendah (PC_L) dan byte tinggi (PC_H) selanjutnya hubungan tersebut seperti hubungan register dengan data bus.

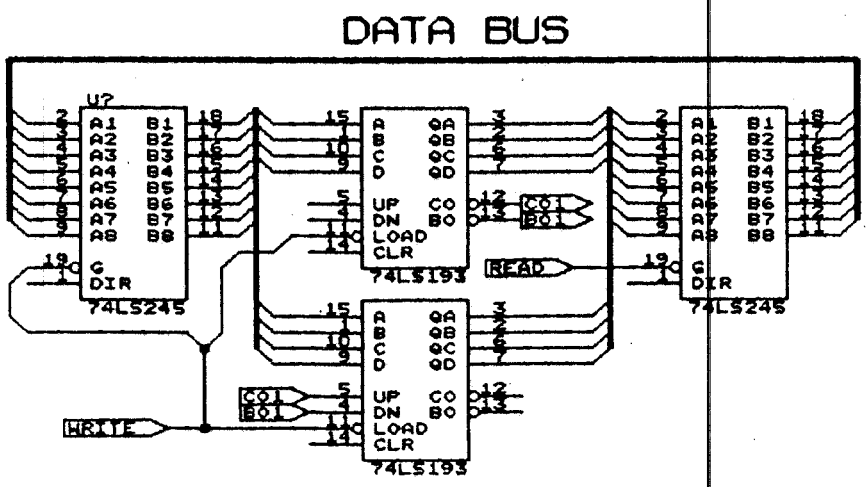


GAMBAR 3.6

TIMING DIAGRAM COUNTER IC 74LS193

III.7. REGISTER

Sebuah flip-flop dapat digunakan untuk menyimpan sebuah bit data. byte dalam processor memerlukan delapan bit data. Susunan bit-bit yang digunakan untuk pemrosesan data dalam processor biasa disebut register. Register merupakan memory internal processor yang sangat berguna untuk pemrosesan data secara cepat, dalam perencanaan ini direncanakan register-register dari IC counter 74LS193 dengan pertimbangan dapat dilakukan proses incremen dan decremen secara langsung lewat pin-pin UP dan DOWN counternya. Bagaimana register dibaca (read) atau ditulisi (write) data, diperlihatkan pada gambar 3.7

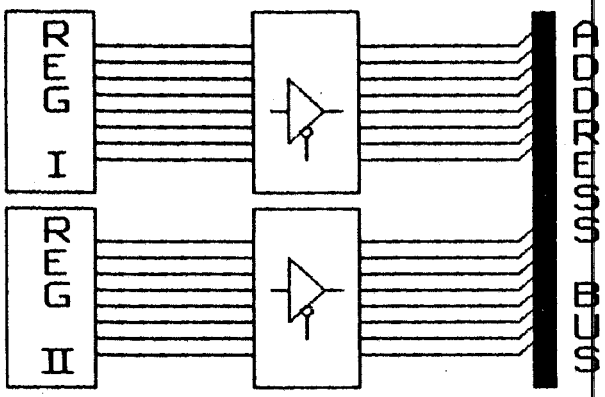


GAMABR 3.7

RANGAKAIAN REGISTER

III.7.1. PASANGAN REGISTER SEBAGAI POINTER

Sering dalam pemrosesan data terdapat proses membaca atau menulis ke memory external dengan alamat yang ditentukan. Untuk mengalamati itu diperlukan register sebagai penunjuk alamat tersebut. Pasangan dua register delapan bit dapat digunakan untuk tujuan di atas. Gambar 3.8 menunjukkan bagaimana susunan pasangan register tersebut dalam hubungannya dengan address bus.



GAMBAR 3.8

PASANGAN REGISTER SEBAGAI POINTER

III.7.2. SHIFT REGISTER

Pemrosesan data digital sering memerlukan proses pergeseran dan perputaran data. Hal ini akan membutuhkan register penggeser atau biasa disebut shift register. Dalam perencanaan ini digunakan IC shift register 74LS195. Dalam proses pergeseran mula-mula register yang akan

digeser disalin ke shift register, kemudian proses pergeseran dilakukan pada shift register, setelah selesai proses, data yang ada pada shift register disalin lagi ke register yang bersangkutan.

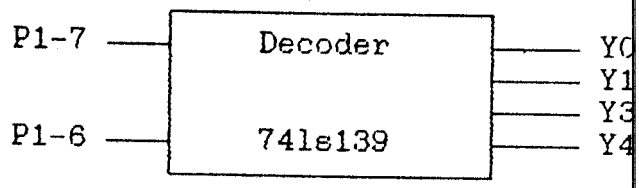
III.8. DECODER INSTRUKSI

Tidak semua instruksi Z80 akan diimplementasikan pada perencanaan ini, instruksi yang akan diimplementasikan terbatas pada program yang dapat menampilkan huruf berjalan pada led matrix 8 X 8. Dari instruksi-instruksi yang terdapat pada program ada yang dapat angsung diterjemahkan dan ada instruksi-instruksi yang membutuhkan mikrocode seperti pada instruksi PUSH.

Setelah data masuk dalam register penangkap maka data tersebut perlu diterjemahkan ke dalam instruksi-instruksi yang sesuai.

Langkah-langkah penerjemahan:

- bit 7 dan 6 dari register P1 diterjemahkan terlebih dahulu dengan decoder 74ls139.



GAMBAR 3.9

PENGKODEAN BIT 7 DAN BIT 6 DARI REGISTER P1

Dari sini Y0,Y1,Y2,Y3 akan dipakai untuk mengaktif-

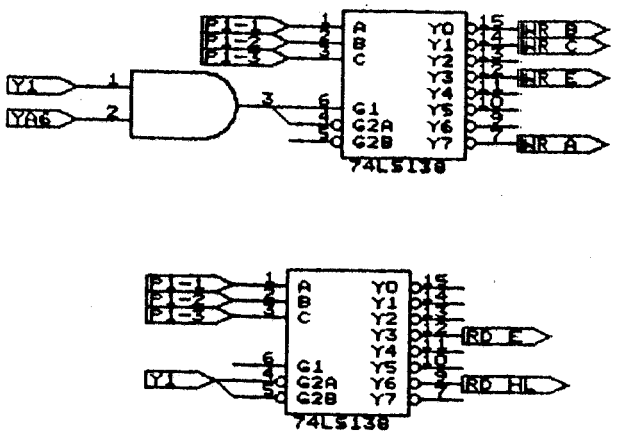
kan decoder-decoder selanjutnya yang sesuai dengan 'op-code'nya.

TABEL 3.3

TABEL KEBENARAN IC 74LS139

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

III.8.1. DECODER INSTRUKSI LD A,E DAN LD A,(HL)



GAMBAR 3.10

DECODER INSTRUKSI LD A,E DAN LD A,(HL)

Instruksi ini mengambil opcode pada bit 7 dan bit 6 adalah 0 dan 1 sehingga dibutuhkan kontrol Y1 untuk mengaktifkan enable dekodernya. Gambar 3.10 menunjukkan bagaimana sebagian dari instruksi-instruksi tersebut diimplementasikan. Sedang untuk Decoder instruksi yang lain dapat dilihat pada lampiran.

TABEL 3.4

TABEL KEBENARAN DECODER 74LS138

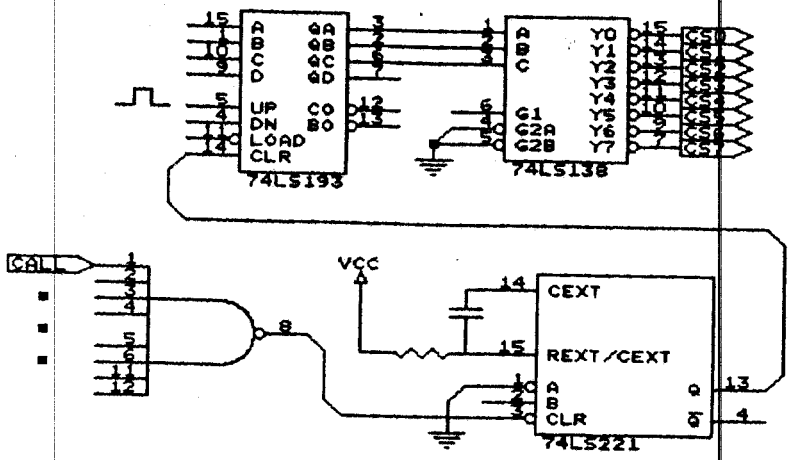
INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	L	L	H	H	H	H	H	H
H	L	L	H	L	L	H	L	H	H	H	H	H
H	L	L	H	H	L	H	H	L	H	H	H	H
H	L	H	L	L	L	H	H	H	L	H	H	H
H	L	H	L	H	L	H	H	H	H	L	H	H
H	L	H	H	L	L	H	H	H	H	H	L	H
H	L	H	H	H	L	H	H	H	H	H	H	L

* G2 = G2A + G2B

III.8.2. INSTRUKSI-INSTRUKSI YANG MEMBUTUHKAN MICROCODE

Penerjemahan instruksi-instruksi yang membutuhkan

mikcrocode memerlukan rangkaian kontrol lagi setelah decoder yang telah dibuat. Lebih jelas dapat dilihat pada rangkaian yang terdapat Gambar 3.11. Dari kontrol micro-code tersebut dilewatkan ke buffer tri-state IC 74LS243 untuk menerjemahkan instruksi-instruksi yang sesuai.



GAMBAR 3.11

RANGKAIAN KONTROL MICROCODE

Berikut akan digambarkan sebagian dari instruksi-instruksi yang membutuhkan microcode. Sedang yang leng-kapnya dapat dilihat pada lampiran.

III.8.2.1. INSTRUKSI CALL NN

Instruksi Call nn membutuhkan step-step/tahap-tahap sebagai berikut:

* PUSH PC yang meliputi tahap-tahap:

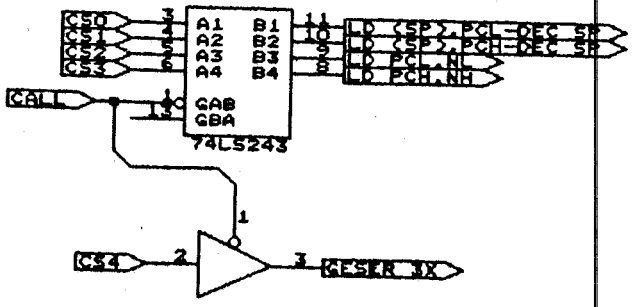
  - LD (SP_L),PC_L

- DEC PC
- LD (SP_H),PC_H
- DEC SP

* JUMP NN yang meliputi tahap-tahap

- LD PC_L,N_L
- LD PC_H,N_H

Gambar 3.12 menunjukkan implementasi dari perintah Call nn.

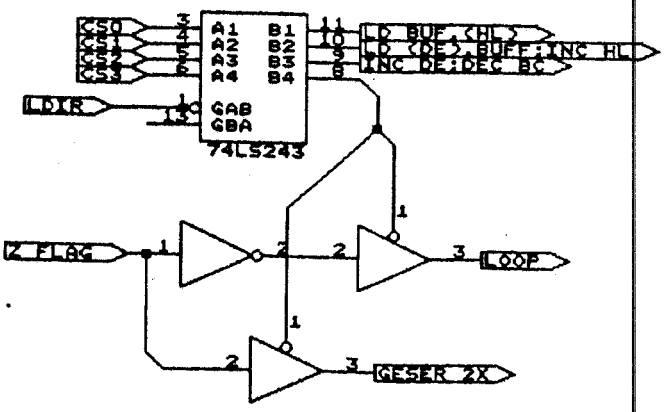


GAMBAR 3.12

TAHAP-TAHAP INSTRUKSI CALL NN

III.8.2.2. INSTRUKSI LDIR

Istruksi LDIR adalah instruksi untuk menyalin dari blok memory ke blok memory yang lain. Blok memory asal alamat awalnya ditunjukkan oleh isi dari pasangan register HL sedang blok memory tujuan alamat awalnya ditunjukkan oleh pasangan register DE. Sebagai counter adalah register BC. Gambar 3.13 menunjukkan implementasi instruksi LDIR.

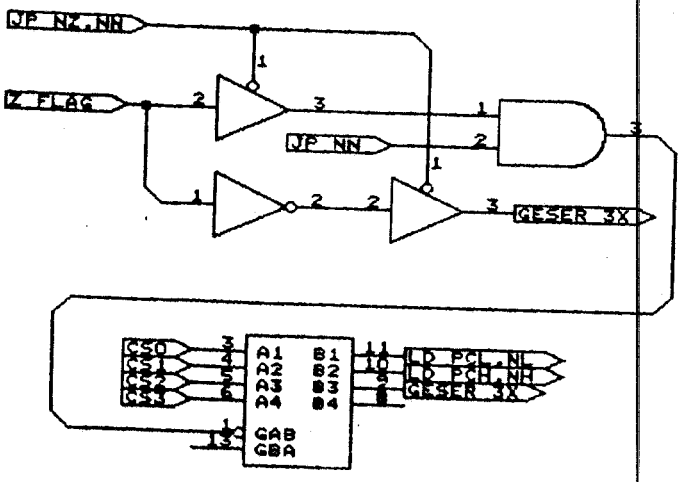


GAMBAR 3.13

TAHAP-TAHAP INSTRUKSI LDIR

III.8.2.3. INSTRUKSI JP NN DAN JP NZ,NN

Gambar 3.14 menunjukkan implentasi intruksi tersebut di atas.

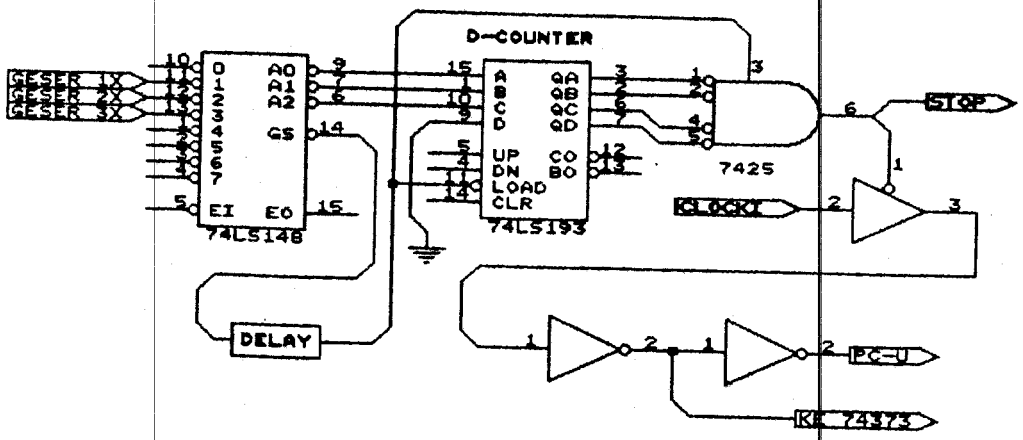


GAMBAR 3.14

TAHAP-TAHAP INSTRUKSI JP NN DAN JP NZ,NN

III.9. RANGAKAIAN PENGHENTI PROGRAM COUNTER

Pada saat register diterjemahkan maka program counter dihentikan sampai operasi tersebut selesai diterjemahkan. Lama penerjemahan tergantung operasi-operasinya. Untuk operasi yang mebutuhkan kode-kode mikro akan memakan waktu yang relatip lebih lama. Rangkaian yang dipakai untk menghenikan PC tersebut dapat dilihat pada gambar 3.15



GAMBAR 3.15

RANGKAIAN PENGHENTI PROGRAM COUNTER

BAB IV
PEMBUATAN DAN PENGUJIAN ALAT

IV.1. PEMBUATAN ALAT

Setelah alat direncanakan maka perlu diwujudkan. Adapun langkah-langkah yang diperlukan adalah:

- Pembuatan artwork dengan menggunakan komputer.
- Setelah pembuatan artwork, hasilnya perlu difilm-kan sehingga dihasilkan klise positip.
- Dibuat PCB yang sesuai dengan klise film tersebut.

IV.2. PNGUJIAN ALAT

Setelah alat dibuat maka perlu diadakan pengujian perbagian dari alat tersebut.

IV.2.1. PENGUJIAN CLOCK INTERNAL

Pengujian clock internal dilakukan dengan cara memberi clock sebesar 2 Mhz. Rangakaian penghenti program counter yang menyetop buffer input clock di'non aktifkan' yaitu dengan cara outputnya selalu dinolkan (dihubungkan dengan ground). Dengan cara tersebut diperoleh clock internal sebesar kurang lebih 870 Khz dengan waktu level high sebesar 350 ns dan waktu level rendah sebesar 800 ns.

IV.2.2. PENGUJIAN PROGRAM COUNTER

Pengujian program counter dilkukan dengan cara yaitu

dengan memberikan clock secara langsung pada input 'up-counter'nya dengan frekuensi yang rendah pada saat permulaan dan ditinggikan perlahan-lahan untuk mengamati perubahan datanya. Output dari program counter tersebut diamati melalui osiloskop. Dalam pengujian ini digunakan generator fungsi yang frekuensi clocknya dapat digeser dari .1 Hz sampai dengan 100 Khz dan juga dicoba pada clock 2 Mhz dari rangkaian clock minimum sistem.

IV.2.3. PENGUJIAN REGISTER

Pengujian register dilakukan dengan cara salah satu register diisi dengan data tertentu lebih dahulu, kemudian register tersebut dibaca (read) pada saat yang sama register lain ditulisi (write). Dalam pengujian ini digunakan register A sebagai register yang pertama yang diisi dengan data 10101010 biner dan register E sebagai register yang kedua. Setelah pengujian maka kedua register berisi data yang sama yaitu 10101010 biner.

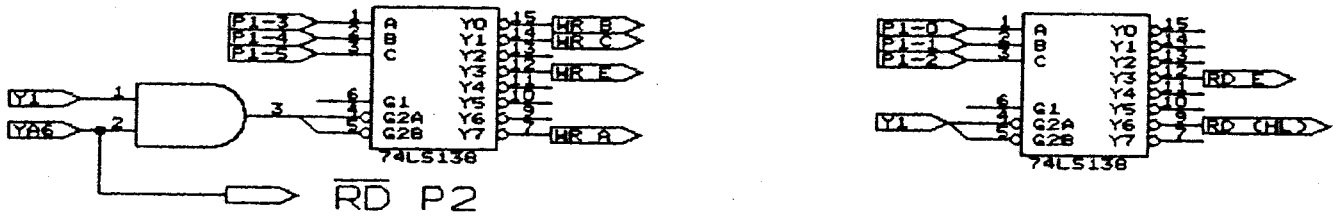
BAB V
KESIMPULAN

Dari hasil perencanaan dan pembuatan unit pemroses komputer digital delapan bit yang telah dibuat dan dari hasil pengujiaan per blok daripada alat dapat diambil beberapa kesimpulan dan saran sebagai berikut:

- Rangkaian clock internal dapat bekerja pada clock input sebesar 2 Mhz, dengan output sesuai yang direncanakan, yaitu mempunyai waktu level high sebesar 360 ns dan waktu level low sebesar minimal 700 ns.
- Program counter yang dibuat telah dicoba dengan clock sampai sebesar 2 Mhz dan dapat bekerja dengan baik.
- Rangkaian register dapat dibuat dari IC counter 74ls193 yang dapat diisi (load) dari register lain atau dibaca untuk disalin ke register lain, dapat pula dilakukan proses incremen dan decremen secara langsung.
- Dalam tugas akhir ini belum bisa dibuat alat sesuai dengan yang direncanakan secara sempurna sehingga perlu pengkajian lebih lanjut untuk menyempurna-kannya.

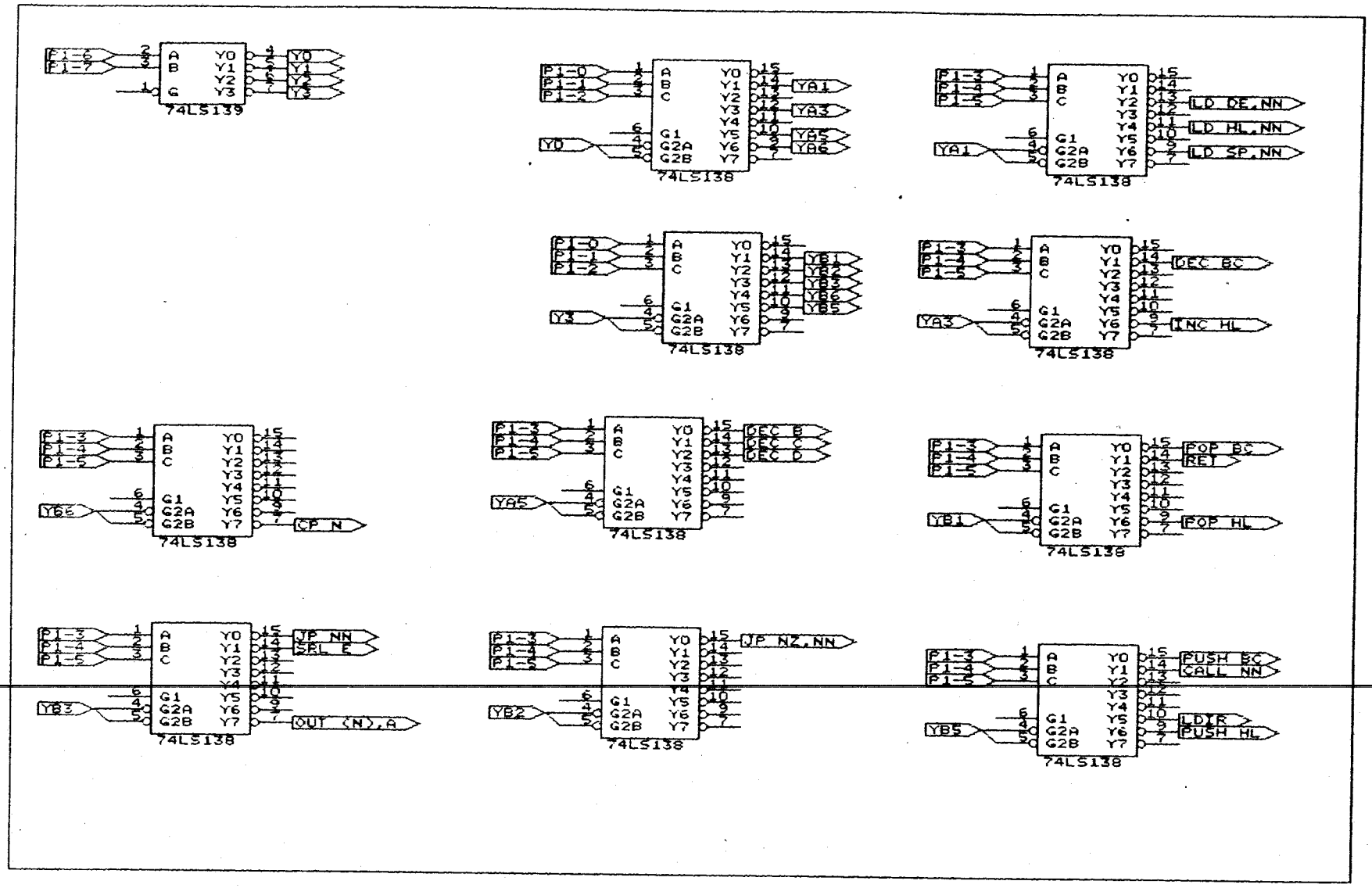
DAFTAR PUSTAKA

1. Avental, Lance A., Z80 ASSEMBLY LANGUAGE PROGRAMMING, McGraw-Hill, Inc., U.S.A., 1979.
2. Carr, Joseph J., Z80 USERS MANUAL, Printice-Hall Company, 1980.
3. Coffron, James W., Z80 APPLICATION, Sybex Inc., Berkeley, 1983.
4. Nashelsky, Louis, INTRODUCTION TO DIGITAL COMPUTER, John and wiley & Sons, Inc., Canada, 1977.
5. Texas Instruments, THE BIPOLAR DIGITAL INTEGRATED CIRCUITS DATA BOOK, Texas Instruments Inc., Texas, 1985.
6. V. Carl Hamacher, Zvonsko G. Vranesic, Safwat G. zaky, COMPUTER ORGANIZATION, McGraw-Hill International Editions, U.S.A., 1984.
7. Zilog, Z80-CPU Z80A-CPU TECHNICAL MANUAL, Zilog Inc., 1977.



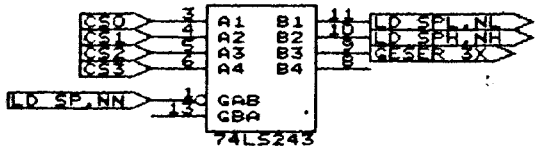
PERINTAH LD R,R' DAN

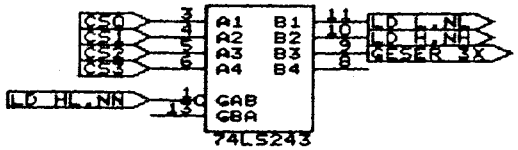
LD R,N

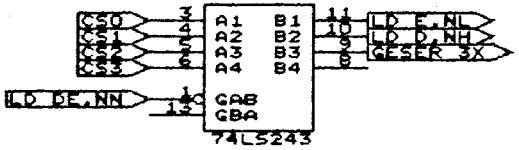

74LS139
74LS138
74LS138
74LS138
74LS138
74LS138
74LS138
74LS138
74LS138
74LS138
74LS138
74LS138
YA1
YA3
YA5
YA6
YB1
YB2
YB3
YB5
YB6
YB7
LD DE,NN
LD HL,NN
LD SP,NN
DEC BC
INC HL
DEC B
DEC C
DEC D
POP BC
RET
POP HL
JP NN
SBC E
OUT (N),A
JP NZ,NN
PUSH BC
CALL NN
LDIR
PUSH HL
CP N

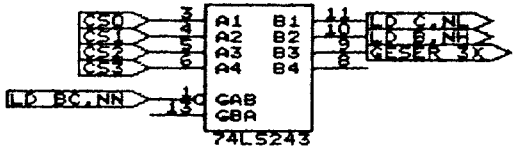
LAMPIRAN

INSTRUKSI YANG MEMBUTUHKAN MICROCODE



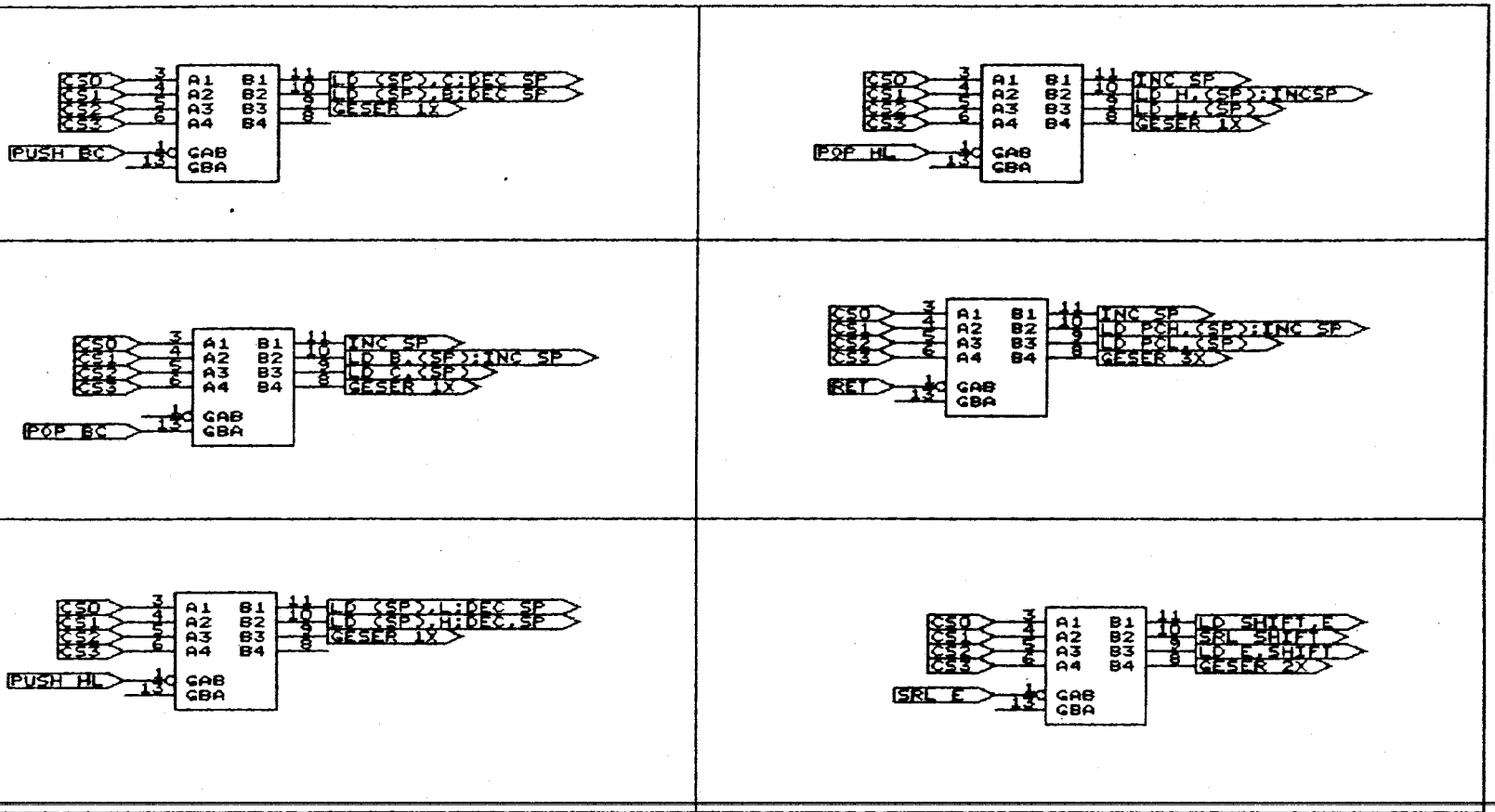






RANGKAIN MICROCODE UNTUK INSTRUKSI:

- LD BC,NN
- LD DE,NN
- LD HL,NN
- LD SP,NN



RANGKAIAN MICROCODE UNTUK

PUSH BC, POP BC, PUSH HL, POP HL

SRL E, DAN RET

LISTING PROGRAM

```
 .z80
ASEG

 ORG                0000H
;
 P8255              EQU        0FH ;Port control dari 8255
 PO_C               EQU        0EH
 PO_B               EQU        0DH
 PO_A               EQU        0CH
 buffer             equ        1001h

;
;POWER UP DELAY
                    LD      B,0
               sum: dec     b
                    Jp      NZ,sum
;INISIALISASI PPI 8255
                    LD      A,10000001B
                    OUT     (P8255),A
;
;INISIALISASI STACK POINTER (SP)

;
;NAMA YANG AKAN DITAMPILKAN
                    LD      SP,17FFH
                    JP      INI
                    DEFB 0EEH
TEXT:   DEFB  'NAMA: ABDUL BASITH '
NOMER:  DEFB ' NRP :2842200200'
        DEFB ' JURUSAN TEKNIK ELEKTRO'
        DEFB ' FAKULTAS TEKNOLOGI INDUSTRI'
        DEFB ' INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA'
        DEFB ' ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789.,:*'
BULET:  DEFB ' *    %'

INI:                LD  hl,TEXT

AWAL:       CALL    SCAN
            LD      a,(hl)
            CP      '%'
            JP      Z,ini
            JP        AWAL

;*************************************************

SCAN:
```

```
          CALL klumpuk
          push hl              ;1
          ld   hl,buffer

          LD C,7
lagi:
          LD b,7                    ;sapuan
ULANG:    push hl              ;2
             push  bc
             LD  E,80H
             LD  c,8

kcol:          ld     a,e
               out    (po_b),a      ;gnd
               srl    e
               ld     a,(hl)
               out    (po_a),a       ; kolom

                  LD     B,0
met:              dec    b            ; ; delay
                  jp     NZ,met

               ld     a,0            ;  reset data di port a
               out    (po_a),a
               inc    hl
               dec    c
               jp     nz,kcol
            pop    bc
            pop    hl             ;-2
            dec    b
            jp     nz,ulang
          inc    hl
          dec    c
          jp     nz,lagi
          pop    hl                  ;-1
     ret

;*****************************************************************

klumpuk:          push    hl
                  CALL   TAMPIL
                   ld     de,buffer
                   ld     b,0
                   ld     c,7
                   ldir
                   pop    hl
                   inc    hl
                   push   hl
                   call   tampil
                   ld     b,0
                   LD     c,7
                   ldir
                   pop    hl
                   ret
```

```
TAMPIL:     ld      a,(hl)
            CP      ' '
            JP      Z,HURUF_SP
            CP      'A'
            JP      Z,HURUF_A
            CP      'B'
            JP      Z,HURUF_B
            CP      'C'
            JP      Z,HURUF_C
            CP      'D'
            JP      Z,HURUF_D
            CP      'F'
            JP      Z,HURUF_F
            CP      'G'
            JP      Z,HURUF_G
            CP      'H'
            JP      Z,HURUF_H
            CP      'I'
            JP      Z,HURUF_I
            CP      'J'
            JP      Z,HURUF_J
            CP      'K'
            JP      Z,HURUF_K
            CP      'L'
            JP      Z,HURUF_L
            CP      'M'
            JP      Z,HURUF_M
            CP      'N'
            JP      Z,HURUF_N
            CP      'O'
            JP      Z,HURUF_O
            CP      'P'
            JP      Z,HURUF_P
            CP      'Q'
            JP      Z,HURUF_Q
            CP      'R'
            JP      Z,HURUF_R
            CP      'S'
            JP      Z,HURUF_S
            CP      'T'
            JP      Z,HURUF_T
            CP      'U'
            JP      Z,HURUF_U
            CP      'V'
            JP      Z,HURUF_V
            CP      'W'
            JP      Z,HURUF_W
            CP      'X'
            JP      Z,HURUF_X
            CP      'Y'
            JP      Z,HURUF_Y
            CP      'Z'
            JP      Z,HURUF_Z
            CP      '0'
```

```
                JP      Z,HURUF_0
                CP      '1'
                JP      Z,HURUF_1
                CP      '2'
                JP      Z,HURUF_2
                CP      '3'
                JP      Z,HURUF_3
                CP      '4'
                JP      Z,HURUF_4
                 CP      '5'
                JP      Z,HURUF_5
                CP      '6'
                JP      Z,HURUF_6
                CP      '7'
                JP      Z,HURUF_7
                CP      '8'
                JP      Z,HURUF_8
                CP      '9'
                JP      Z,HURUF_9
                CP      '.'
                JP      Z,HURUF_11
                CP      ','
                JP      Z,HURUF_12
                CP      ':'
                JP       Z,HURUF_13
                CP      '*'
                JP      Z,HURUF_15
                CP      25H
                JP      Z,HURUF_20

HURUF_SP:   LD hl,HUR_SP
            RET
HURUF_A:    LD hl,HUR_A
            RET
HURUF_B:    LD hl,HUR_B
            RET
HURUF_C:    LD hl,HUR_C
            RET
HURUF_D:    LD hl,HUR_D
            RET
HURUF_E:    LD hl,HUR_E
            RET
HURUF_F:    LD hl,HUR_F
            RET
HURUF_G:    LD hl,HUR_G
            RET
HURUF_H:    LD hl,HUR_H
            RET
HURUF_I:    LD hl,HUR_I
            RET
HURUF_J:    LD hl,HUR_J
            RET
HURUF_K:    LD hl,HUR_K
            RET
HURUF_L:    LD hl,HUR_L
```

```
            RET
HURUF_M:    LD hl,HUR_M
            RET
HURUF_N:    LD hl,HUR_N
            RET
HURUF_O:    LD hl,HUR_O
            RET
HURUF_P:    LD hl,HUR_P
            RET
HURUF_Q:    LD hl,HUR_Q
            RET
HURUF_R:    LD hl,HUR_R
            RET
HURUF_S:    LD hl,HUR_S
            RET
HURUF_T:    LD hl,HUR_T
            RET
HURUF_U:    LD hl,HUR_U
            RET
HURUF_V:    LD hl,HUR_V
            RET
HURUF_W:    LD hl,HUR_W
            RET
HURUF_X:    LD hl,HUR_X
            RET
HURUF_Y:    LD hl,HUR_Y
            RET
HURUF_Z:    LD hl,HUR_Z
            RET
HURUF_0:    LD hl,HUR_0
            RET
HURUF_1:    LD hl,HUR_1
            RET
HURUF_2:    LD hl,HUR_2
            RET
HURUF_3:    LD hl,HUR_3
            RET
HURUF_4:    LD hl,HUR_4
            RET
HURUF_5:    LD hl,HUR_5
            RET
HURUF_6:    LD hl,HUR_6
            RET
HURUF_7:    LD hl,HUR_7
            RET
HURUF_8:    LD hl,HUR_8
            RET
HURUF_9:    LD hl,HUR_9
            RET
HURUF_11:    LD hl,HUR_11
            RET
HURUF_12:    LD hl,HUR_12
            RET
HURUF_13:    LD hl,HUR_13
            RET
```

```
HURUF_15:   LD hl,HUR_15
            RET
HURUF_20:   LD hl,HUR_20
            RET


;************************************


HUR_SP:    DEFB 00H,00H,00H,00H,00H,00H,00H
HUR_A:     DEFB 0H,0F8H,25H,022H,025H,0F8H,0H
HUR_B:      DEFB 0H,0FFH,92H,92H,92H,6CH,0H
HUR_C:     DEFB 0H,7CH,82H,82H,82H,44H,00H
HUR_D:    DEFB 0H,0FFH,82H,82H,82H,7CH,0H
HUR_E:     DEFB 0H,0FFH,92H,92H,82H,44H,0H
HUR_F:      DEFB 0H,0FFH,12H,12H,02H,04H,0H
HUR_G:        DEFB 0H,07CH,82H,92H,92H,74H,0H
HUR_H:         DEFB 0H,0FFH,10H,10H,10H,0FFH,0H
HUR_I:      DEFB 0H,0H,82H,0FFH,82H,0H,0H
HUR_J:        DEFB 0H,40H,80H,80H,7EH,0H,0H
HUR_K:       DEFB 0H,0FFH,20H,10H,28H,0C7H,0H
HUR_L:         DEFB 0H,0FFH,80H,80H,80H,40H,0H
HUR_M:       DEFB 0H,0FFH,4H,8H,4H,0FFH,0H
HUR_N:          DEFB 0H,0FFH,4H,8H,01H,0FFH,0H
HUR_O:       DEFB 0H,7CH,82H,82H,82H,7CH,0H
HUR_P:      DEFB 0H,0FFH,12H,12H,12H,0CH,0H
HUR_Q:      DEFB 0H,0FFH,92H,0A2H,42H,0BFH,0H
HUR_R:       DEFB 0H,0FFH,12H,32H,52H,8CH,0H
HUR_S:       DEFB 0H,4CH,92H,92H,92H,64H,0H
HUR_T:      DEFB 0H,6H,02H,0FFH,02H,6H,0H
HUR_U:      DEFB 0H,7FH,80H,80H,80H,7FH,0H
HUR_V:      DEFB 0H,3FH,40H,80H,40H,3FH,0H
HUR_W:      DEFB 0H, 0FFH,40H,20H,40H,0FFH,0H
HUR_X:      DEFB 0H,0C6H,28H,10H,28H,0C6H,0H
HUR_Y:      DEFB 0H,06H,08H,10H,08H,06H,0H
HUR_Z:      DEFB 0H,0C6H,20H,10H,08H,0C6H,0H
HUR_0:      DEFB 0H,07CH,0A2H,92H,88H,7CH,0H
HUR_1:      DEFB 0H,00H,84H,0FFH,84H,0H,0H
HUR_2:        DEFB 0H,0C4H,0A2H,92H,88H,4CH,0H
HUR_3:    DEFB 0H,42H,92H,9AH,96H,62H,0H
HUR_4:    DEFB 0H,20H,30H,28H,24H,0FFH,0H
HUR_5:    DEFB 0H,4EH,8AH,8AH,8AH,72H,0H
HUR_6:    DEFB 0H,70H,98H,94H,92H,60H,0H
HUR_7:    DEFB 0H,6H,0E2H,12H,0AH,06H,0H
HUR_8:    DEFB 0H,6CH,92H,92H,92H,6CH,0H
HUR_9:    DEFB 0H,0CH,92H,52H,32H,1CH,0H
HUR_11:    DEFB 0H,0H,0C0H,0C0H,00H,00H,0H
HUR_12:    DEFB 0H,0H,80H,40H,48H,10H,0H
HUR_13:    DEFB 0H,0H,66H,66H,00H,0H,0H
HUR_15:    DEFB 0H,56H,38H,0FFH,38H,56H,0H
HUR_20:   DEFB '%'
end
```