UAD

MODUL PRAKTIKUM

ELEKTRONIKA KOMPUTER DIGITAL

Prof. Dr. Muchlas, M.T.

Laboratorium Dasar Elektro

2024

PETUNJUK PRAKTIKUM
ELEKTRONIKA KOMPUTER DIGITAL

Disusun oleh:

Dr. H. Muchlas, M.T.

Arsyad Cahya Subrata, S.T., M.T.

Ahmad Raditya Cahya Baswara, S.T., M.Eng.

LABORATORIUM DASAR TEKNIK ELEKTRO

PROGRAM STUDI TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS AHMAD DAHLAN

2024

SEJARAH REVISI

REVISI KE	TANGGAL REVISI	URAIAN REVISI
1	10 September 2022	Penyesuaian tema pengerjaan sehingga menjadi 10 unit.
2	17 September 2023	Penambahan tugas pendahuluan dan laporan sememtara.
3	15 September 2024	Penambahan estimasi waktu praktikum dan petunjuk K3.

Waktu Praktikum

Pelaksanaan Waktu praktikum di laboratorium Teknik Elektro berlangsung selama 180 menit dengan pembagian waktu diatur sebagai berikut:

1. Tugas Pendahuluan/Pre Test : 20 Menit
2. Percobaan Praktikum : 100 Menit
3. Laporan Praktikum : 60 Menit

Keselamatan

Pada prinsipnya, untuk mewujudkan praktikum yang aman diperlukan partisipasi seluruh praktikan dan asisten pada praktikum yang bersangkutan. Dengan demikian, kepatuhan setiap praktikan terhadap uraian panduan pada bagian ini akan sangat membantu mewujudkan praktikum yang aman.

Bahaya Listrik

Perhatikan dan pelajari tempat-tempat sumber listrik (stop-kontak dan circuit breaker) dan cara menyala-matikannya. Jika melihat ada kerusakan yang berpotensi menimbulkan bahaya, laporkan pada asisten.

1. Hindari daerah atau benda yang berpotensi menimbulkan bahaya listrik (sengatan listrik) secara tidak disengaja, misalnya kabel jala-jala yang terkelupas dll.
2. Tidak melakukan sesuatu yang dapat menimbulkan bahaya listrik pada diri sendiri atau orang lain.
3. Keringkan bagian tubuh yang basah karena, misalnya, keringat atau sisa air wudhu.
4. Selalu waspada terhadap bahaya listrik pada setiap aktivitas praktikum.

Kecelakaan akibat bahaya listrik yang sering terjadi adalah tersengat arus listrik. Berikut ini adalah hal-hal yang harus diikuti praktikan jika hal itu terjadi:

1. Jangan panik,
2. Matikan semua peralatan elektronik dan sumber listrik di meja masing-masing dan di meja praktikan yang tersengat arus listrik,
3. Bantu praktikan yang tersengat arus listrik untuk melepaskan diri dari sumber listrik,
4. Beritahukan dan minta bantuan asisten, praktikan lain dan orang di sekitar anda tentang terjadinya kecelakaan akibat bahaya listrik.

DFTAR ISI

PERCOBAAN I
TRANSFER REGISTER

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat:

1. Menuliskan mikrooperasi transfer register
2. Mengimplementasikan hardware dari mikrooperasi transfer register

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

1. Jelaskan perbedaan yang menonjol dari transfer Paralel dan transfer Seri!
2. Jelaskan proses transfer Bus dan transfer Memori!

C. TEORI

Operasi yang berhubungan dengan data yang tersimpan di dalam register atau flip-flop dinamakan mikrooperasi (*microoperation*) seperti *load*, *clear*, *shift*, dan*rotate*. *Load* adalah operasi untuk memuati atau mengisi data ke dalam register, *clear* merupakan operasi menghapus data dalam register, *shift* atau geser adalah operasi untuk menggeser posisi data dalam register ke kiri atau ke kanan, dan *rotate* merupakan operasi untuk memutar data ke kiri atau ke kanan. Selain itu, terdapat pulamikrooperasi aritmetika seperti penambahan, pengurangan, perkalian, pembagian, *increment* (penambahan dengan 1) dan *decrement* (pengurangan dengan 1) terhadap isi suatu register, serta operasi mikro logika seperti AND, OR, dan NOT.

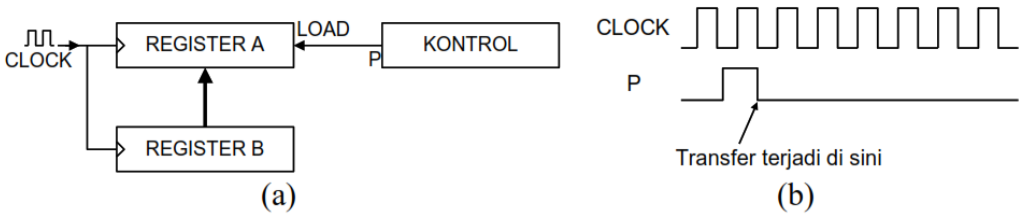
Kecuali dapat dioperasikan dengan berbagai mikrooperasi seperti di atas, data yang tersimpan di dalam register juga dapat dipindah dari satu register ke register yang lain melalui operasi transfer. Pada operasi ini, isi suatu register yang dipindah ke register lain, setelah operasi dilakukan keadaannya tetap atau tidak berubah. Dengan kata lain, operasi transfer merupakan proses penyalinan data. Dalam hal ini, register yang isinya disalin dinamakan register sumber (*source register*) dan register penampung data salinan dinamakan register tujuan (*destination register*). Mekanisme transfer data dapat dilakukan dengan berbagai cara antara lain transfer paralel, transfer seri, transfer bus, dan transfer memori.

1. Transfer Paralel

Pada transfer data paralel, pemindahan atau penyalinan data dari register sumber ke register tujuan dilaksanakan secara serempak. Artinya, semua data yang tersimpan pada setiap flip-flop yang merupakan elemen-elemen register sumber disalin secara serempak ke register tujuan. Mikrooperasi transfer paralel dinyatakan sebagai berikut:

$$P: A \leftarrow B$$

Artinya, jika ada sinyal P, data dari register B disalin ke register A. Implementasi hardware dan timing diagram dari pernyataan tersebut adalah:



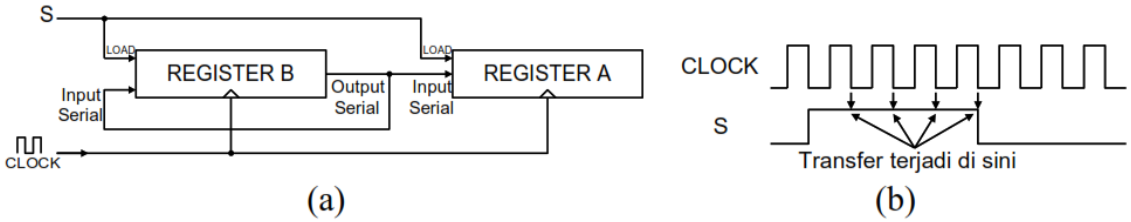
Gambar 1. Transfer paralel P: A ← B: (a) implementasi *hardware* (b) timing diagram

2. Transfer Seri

Pada transfer data seri, pemindahan data dilakukan bit demi bit. Untuk menyelenggarakan proses ini diperlukan register geser atau register seri. Transfer ini juga memerlukan operasi *rotate* atau putar sehingga output LSB register sumber (B_0) selain dihubungkan ke input MSB register tujuan (A_3), juga diumpankan ke inputnya sendiri yakni input MSB (B_3). Untuk register 4-bit, mikrooperasi transfer seri dapat ditulis:

$$S: A_3 \leftarrow B_0,\ B_3 \leftarrow B_0,\ A_i \leftarrow A_{i+1},\ B_i \leftarrow B_{i+1} \quad i=0,1,2$$

Implementasi *hardware* dan *timing diagram* untuk transfer seri dengan register 4-bit ditunjukkan pada Gambar 2.



Gambar 2. Transfer seri S: $A_3 \leftarrow B_0,\ B_3 \leftarrow B_0,\ A_i \leftarrow A_{i+1},\ B_i \leftarrow B_{i+1}$ i=0,1,2
(a) implementasi *hardware*, (b) *timing diagram*

3. Transfer Bus

Transfer bus dilakukan melalui dua tahap, yakni transfer dari register sumber ke bus, dan transfer dari bus ke register tujuan. Untuk dua buah register sumber yakni A dan B serta empat buah register tujuan yakni R0, R1, R2, dan R3, mikrooperasi kedua tahap tersebut adalah sebagai berikut.

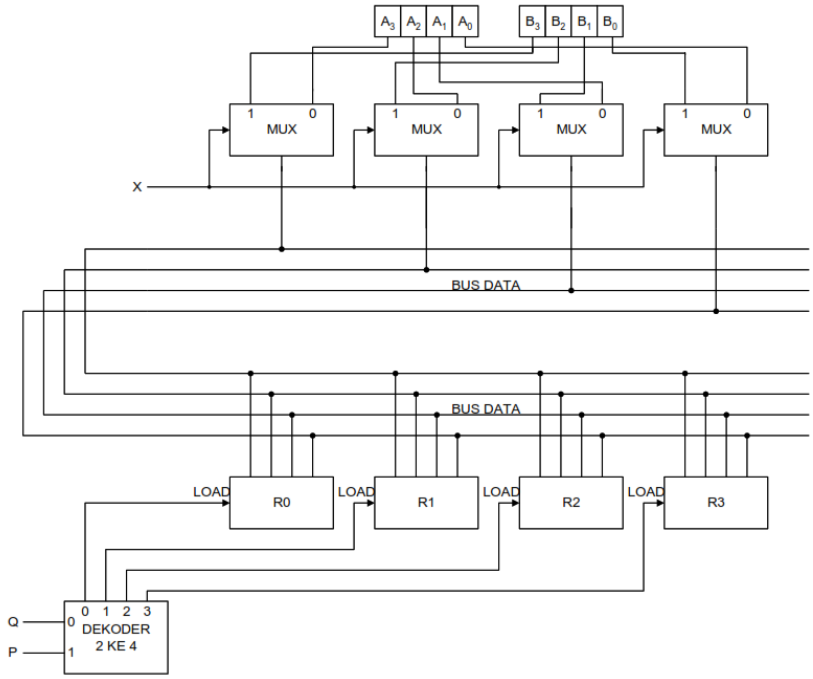
Transfer dari register sumber ke bus:

$\bar{X}$: BUS ← A
X : BUS ← B

Transfer dari bus ke register tujuan:

$\bar{P}\bar{Q}$: R0 ← BUS
$\bar{P}Q$: R1 ← BUS
$P\bar{Q}$: R2 ← BUS
PQ : R3 ← BUS

Implementasi *hardware* dari kedua mikrooperasi tersebut adalah sebagai berikut.



Gambar 3. Implementasi *hardware* dari transfer bus untuk register 4-bit

4. Transfer Memori

Pada transfer memori terdapat dua operasi yakni operasi baca (read) dan operasi tulis (*write*). Transfer memori melibatkan dua buah register yakni MAR (*memory address register*) dan MBR (*memory buffer register*). MAR merupakan register yang berisi kode alamat/lokasi memori yang akan dibaca/ditulisi, dan MBR merupakan register yang berfungsi menampung data hasil pembacaan pada operasi baca atau data yang akan disimpan ke memori pada operasi tulis.

Operasi baca merupakan penyalinan data dari lokasi/alamat memori yang ditunjuk oleh isi register MAR ke register MBR. Sedangkan operasi tulis merupakan penyalinan data dari MBR ke lokasi/alamat memori yang ditunjuk oleh isi register MAR. Mikrooperasi untuk kedua operasi itu dapat ditulis sebagai berikut.

Mikrooperasi baca memori:

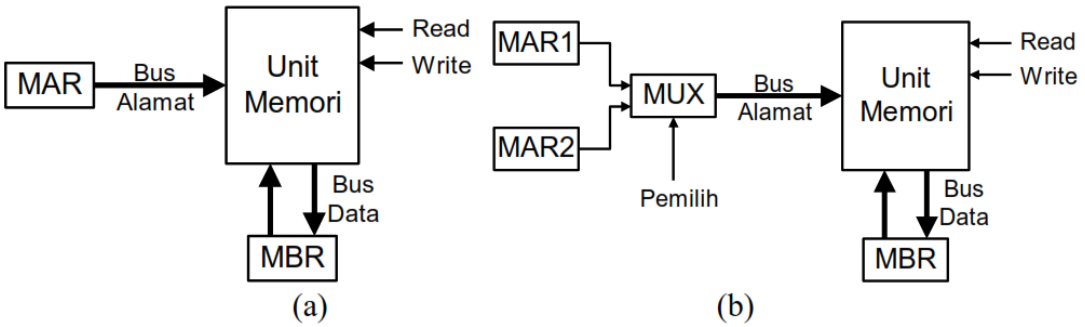
Read: MBR ← M[MAR]

Mikrooperasi tersebut dapat diartikan jika ada sinyal kontrol *read*, maka data yang ada pada alamat memori yang ditunjuk oleh isi register MAR disalin ke register MBR. Jika isi register MAR adalah 03FC heksadesimal, maka mikrooperasi tersebut akan menyebabkan data pada alamat memori 03FC heksadesimal disalin ke register MBR. Mikrooperasi tulis memori:

Write: M[MAR] ← MBR

Kebalikan dari operasi baca, pada operasi tulis jika ada sinyal kontrol *write* maka data yang ada pada MBR disalin ke alamat memori yang lokasinya ditunjuk oleh MAR.

Implementasi *hardware* dari kedua mikrooperasi itu ditunjukkan pada gambar 4 berikut ini.



Gambar 4. Implementasi *hardware* mikrooperasi transfer memori:
(a) menggunakan MAR tunggal, (b) MAR ganda

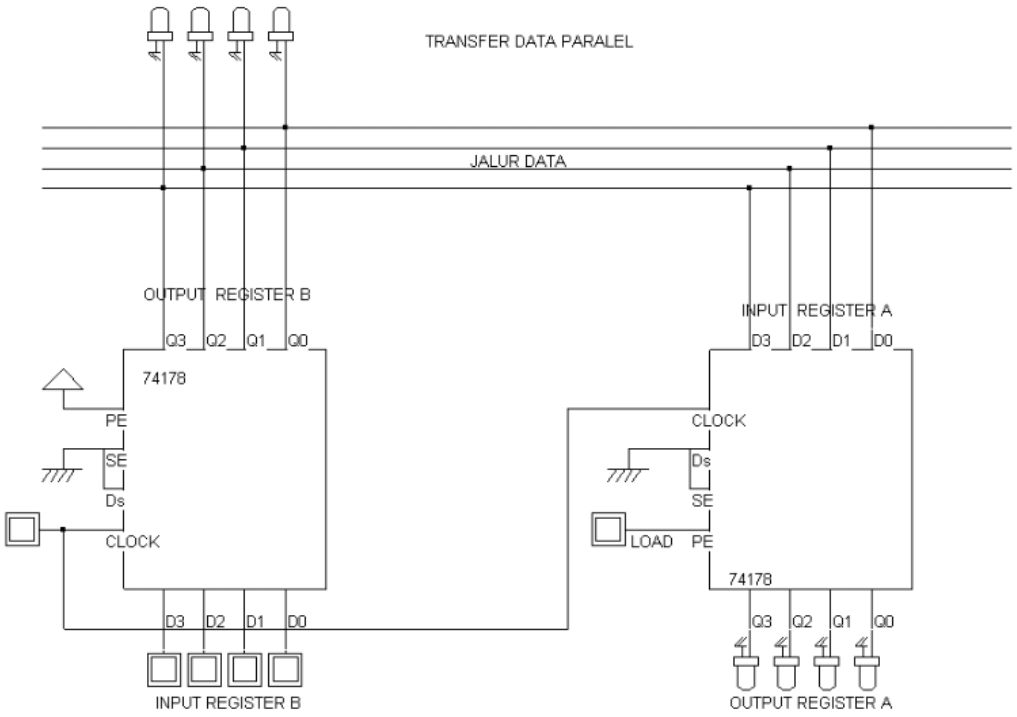
D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. Software Editor dan Simulator Logika DSCH2
3. Modul TRANSFER_PARALEL.SCH, modul TRANSFER_SERI.SCH, modul TRANSFER_BUS.SCH, dan modul TRANSFER_MEMORI.SCH.

E. PROSEDUR PERCOBAAN

Transfer Paralel

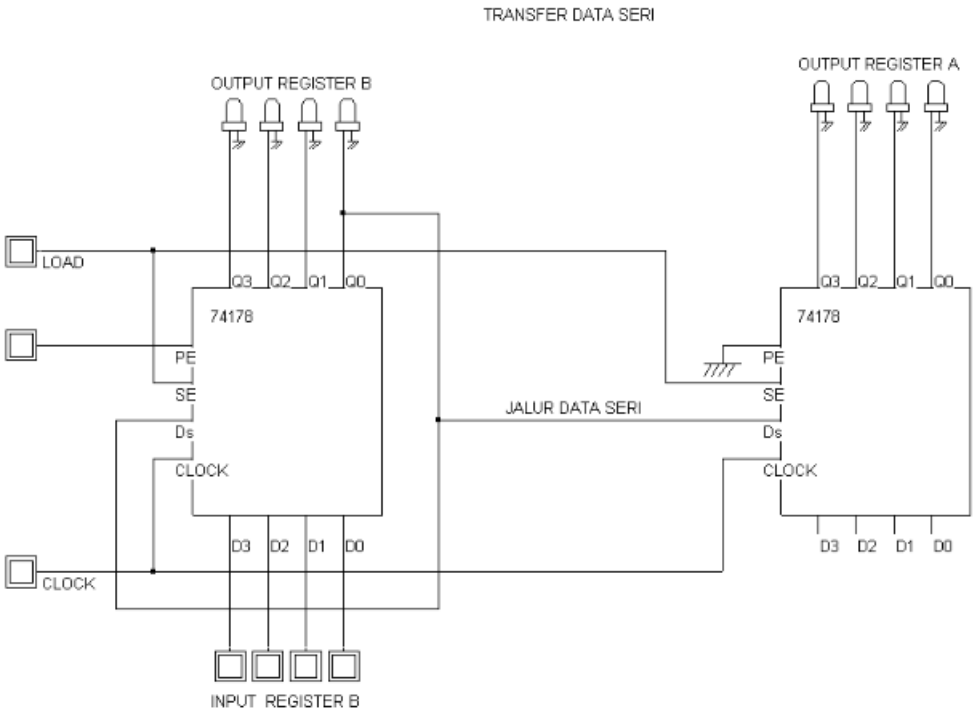
1. Jalankan DSCH2 dan load file TRANSFER_PARALEL.SCH. Pastikan anda memperoleh tampilan layar monitor sebagai berikut:
2. Diumpamakan anda akan melakukan transfer paralel data 1001 dari register B ke register A. Klik *run simulation*, berikan input register B dengan data $D_3D_2D_1D_0$=1001, berikan sebuah sinyal *clock* dengan menekan tombol CLOCK dua kali sehingga register B terisi data 1001 atau 9 desimal.
3. Berikan LOAD=1 pada register A dan sebuah sinyal CLOCK. Apakah data pada register B telah disalin ke register A?
4. Ulangi percobaan dengan berbagai data pada register B! Coba berikan LOAD=0 pada register A, apakah data dapat ditransfer?



Gambar 5. Tampilan DSCH2 untuk transfer paralel

Transfer Seri

1. Dari layar editor DSCH2, load file TRANSFER_SERI.SCH sehingga anda memperoleh tampilan seperti pada gambar 6.

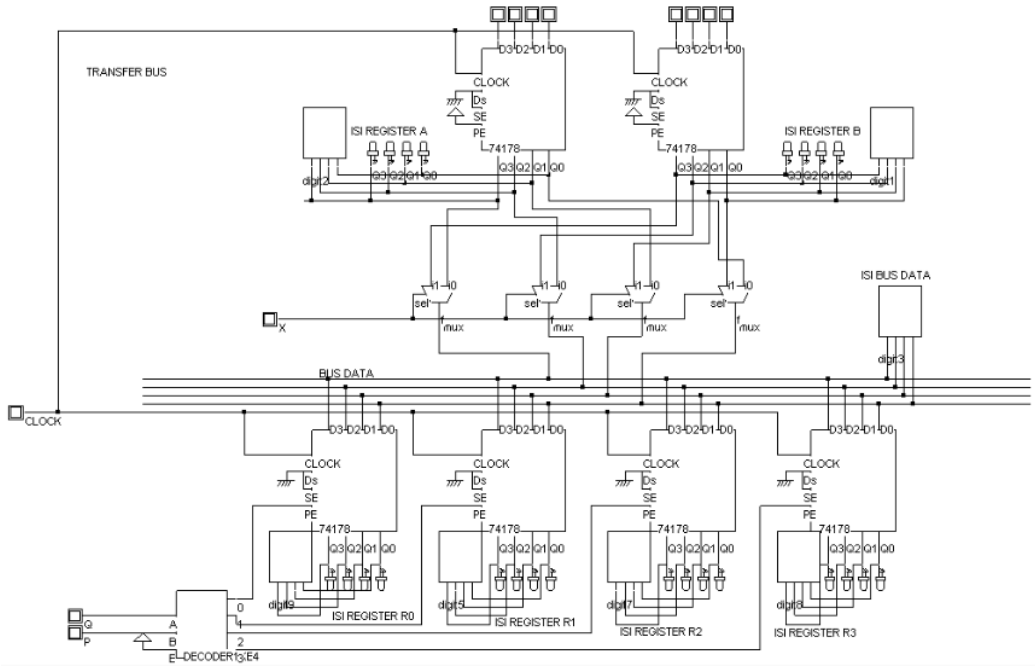


Gambar 6. Tampilan DSCH2 untuk transfer seri

2. Diumpamakan anda akan melakukan transfer seri data 1001 dari register B ke register A. Klik *run simulation*, berikan input register B dengan data $D_3D_2D_1D_0$=1001 dan PE=1, serta sebuah sinyal CLOCK sehingga register B terisi data 1001 atau 9 desimal.
3. Berikan LOAD=1, dan matikan sinyal PE pada register B. Berikan empat buah sinyal CLOCK, amati isi register A! Apakah sudah sama dengan isi register B?
4. Ulangi percobaan untuk berbagai data pada register B!

Transfer Bus

1. Dari layar editor DSCH2, load file TRANSFER_BUS.SCH sehingga anda memperoleh tampilan seperti pada gambar 7.

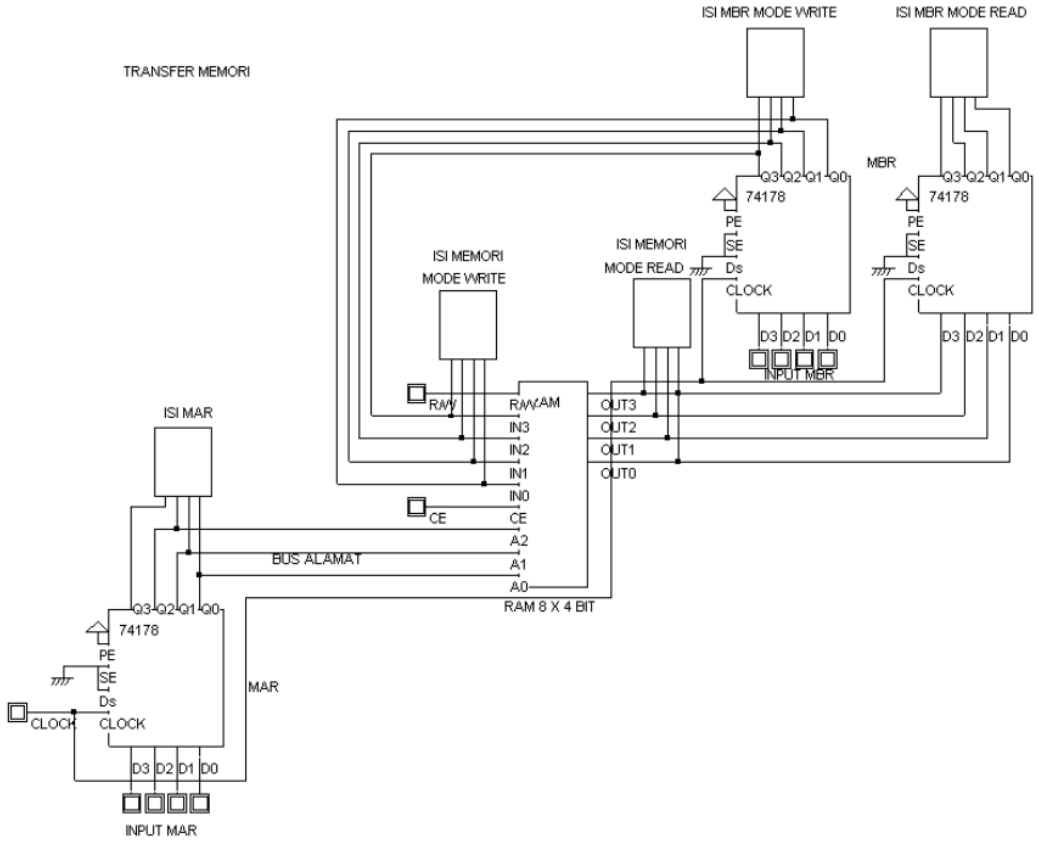


Gambar 7. Tampilan DSCH2 untuk transfer bus

2. Diumpamakan pada register A berisi data 0101 dan pada register B berisi 1001. Data pada register A akan ditransfer ke register R0 dan data pada register B ditransfer ke register R2. Klik *run simulation*, berikan input register A dengan data $D_3D_2D_1D_0$=0101 dan input register B dengan $D_3D_2D_1D_0$=1001. Berikan X=0 dan sebuah sinyal CLOCK, apakah isi register A telah disalin ke bus data?
3. Berikan PQ=00 pada *decoder* dan sebuah sinyal CLOCK, apakah data pada bus data telah dapat ditransfer ke register R0?
4. Selanjutnya berikan X=1 sehingga isi register B ditransfer ke bus data. Berikan PQ=10 dan sebuah sinyal CLOCK. Apakah isi bus data telah dapat ditransfer ke register R2?

Transfer Memori

1. Load file TRANSFER_MEMORI.SCH sehingga diperoleh tampilan layar monitor seperti pada gambar 8.



Gambar 8. Tampilan DSCH2 untuk transfer memori

2. Memori yang digunakan pada percobaan ini adalah RAM dengan ukuran 8 x 4-bit, artinya RAM tersebut mampu menampung data sebanyak 8 alamat/lokasi dengan panjang data masing-masing 4-bit. RAM dilengkapi dengan pin pengontrol CE dan R/W. Pin CE (*chip enable*) digunakan untuk mengaktifkan *chip* RAM, karena jenisnya *active-high* maka untuk mengaktifkan RAM pin CE harus diberi sinyal tinggi (CE=1). R/W merupakan pin untuk memilih mode operasi RAM. Agar RAM beroperasi pada mode tulis (*write*), R/W diberi sinyal rendah (R/W=0), dan untuk mengatur agar RAM bekerja pada mode baca (*read*), R/W diberi sinyal tinggi (R/W=1).
3. Diumpamakan anda akan melakukan operasi tulis (*write*) ke dalam memori dengan data sebagai berikut:

Tabel 1. Data yang akan diisikan ke RAM

ALAMAT	DATA BINER				DATA HEKSADESIMAL
	D_3	D_2	D_1	D_0	
0	1	0	1	0	A
1	1	0	1	1	B
2	1	1	0	0	C
3	1	1	0	1	D
4	1	1	1	0	E
5	1	1	1	1	F
6	0	0	0	1	1
7	0	0	1	0	2

Operasi Write:

1. **Menulisi alamat 0:** Berikan input data 0000 pada MAR diikuti dengan pemberian sebuah sinyal CLOCK sehingga pada bus alamat terdapat kode alamat 0000. Berikan sinyal CE=0 dan R/W=0 agar RAM bekerja pada mode *write*. Berikan input MBR dengan data 1010 atau A heksadesimal diikuti dengan pemberian sebuah sinyal CLOCK sehingga pada input RAM terdapat data 1010.
2. Untuk menulisi alamat yang lain, lakukan prosedur 4 dengan mengganti isi MAR dengan kode alamat penyimpanan dan mengganti isi MBR dengan data yang akan disimpan sesuai tabel 1.

Operasi Read:

1. Setelah operasi tulis/*write* tersebut, saat ini RAM telah terisi dengan data sesuai tabel 1.
2. Coba lakukan pembacaan isi RAM tersebut. Berikan sinyal CE=0 dan R/W=1 agar RAM bekerja pada mode baca/*read*.
3. Isilah register MAR dengan data 0000, berikan clock sekali sehingga pada bus alamat terdapat data 0000. Berikan clock sekali untuk menempatkan isi lokasi 0 ke MBR. Apakah MBR sudah berisi data alamat 0?
4. Lakukan pembacaan alamat ke-1, ke-2, ke-3, ke-4, ke-5, ke-6, dan ke-7. Apakah isinya sesuai dengan tabel 1?

F. TUGAS AKHIR

1. Apa fungsi LOAD pada suatu register! Pada operasi apa kontrol LOAD diperlukan dan pada operasi apa tidak diperlukan?
2. Jelaskan keunggulan dan kelemahan transfer paralel dibandingkan transfer seri?
3. Pada transfer bus yang mengandung 4 buah register sumber dengan panjang 4-bit dan 8 buah register tujuan, sebutkan spesifikasi dan jumlah MUX yang diperlukan! Sebutkan pula spesifikasi decoder yang diperlukan!
4. Tuliskan mikrooperasi dan gambarkan implementasi *hardware* dari transfer bus pada tugas nomor 3!

5. Kemukakan fungsi register MBR dan MAR pada transfer memori! Berapa panjang register MAR untuk membaca/menulisi memori dengan kapasitas 16 byte,64 byte, 1KB, 8KB, dan 64KB?

PERCOBAAN II
RANGKAIAN ALU (ARITHMETIC LOGIC UNIT)

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat:

1. Merancang rangkaian ALU sederhana.
2. Menjelaskan cara kerja ALU

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

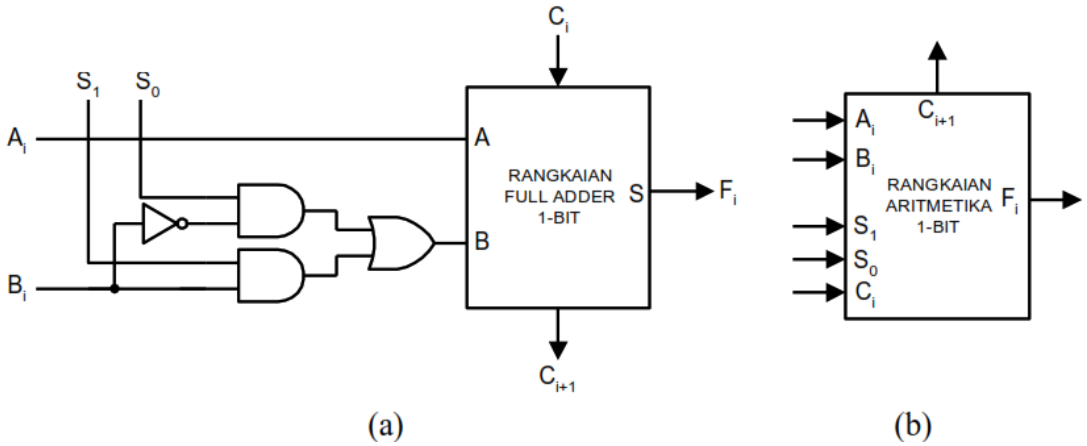
1. Jelaskan bagaimana ALU bekerja!
2. Jelaskan fungsi utama ALU dalam computer!

C. TEORI

Mikrokomputer didefinisikan sebagai sistem interkoneksi antara mikroprosesor, unit penyimpan dan antarmuka input/output. Definisi ini mengacu pada konsep arsitektur mesin Von Neumann yang merupakan dasar bagi pengembangan mesin-mesin komputer modern. Sedangkan mikroprosesor diartikan sebagai suatu *chip* yang di dalamnya terkandung rangkaian *arithmetic logic unit* (ALU), rangkaian *control unit* (CU) dan beberapa register. Rangkaian ALU memegang peranan yang amat penting dalam keseluruhan kerja sistem mikrokomputer, karena bagian ini mengemban tugas yang paling utama yakni melakukan proses komputasi.

Rangkaian Aritmetika

Secara umum ALU terdiri atas rangkaian aritmetika dan rangkaian logika. Rangkaian aritmetika dibangun dari rangkaian *full adder* yang dilengkapi dengan beberapa rangkaian pemilih fungsi. Kemampuan ALU dalam melaksanakan fungsi komputasi sangat tergantung dari kerumitan rangkaian pemilih fungsi ini. Pada contoh berikut akan ditunjukkan rangkaian aritmetika yang mampu melaksanakan 7 operasi aritmetika. Untuk membangun rangkaian aritmetika n-bit, dimulai dari rangkaian aritmetika 1-bit.



Gambar 9. Rangkaian aritmetika 1 bit: (a) rangkaian, (b) simbol

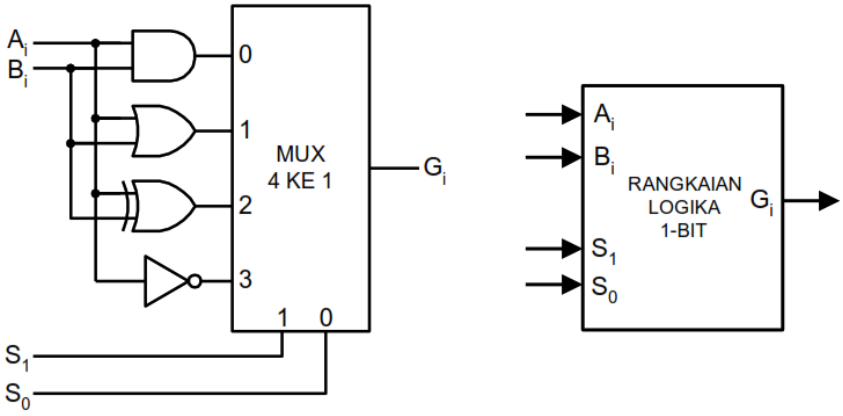
Dengan menggunakan rangkaian pada gambar 9, rangkain aritmetika tersebut dapat melaksanakan operasi seperti pada tabel fungsi berikut ini.

Tabel 2. Tabel fungsi rangkaian aritmetika

PEMILIH FUNGSI			OUTPUT	NAMA OPERASI
S_1	S_0	C_i	F	
0	0	0	F=A	Transfer A
0	0	1	F=A+1	Increment A
0	1	0	F=A+not B	Penjumlahan A dengan not B
0	1	1	F=A-B	Pengurangan
1	0	0	F=A+B	Penjumlahan
1	0	1	F=A+B+1	Penjumlahan dengan carry
1	1	0	F=A-1	Decrement A
1	1	1	F=A	Transfer A

1. Rangkaian Aritmetika

Selain rangkaian aritmetika, dalam ALU terdapat pula rangkaian logika. Rangkaian ini merupakan gabungan dari beberapa gerbang logika dasar dan rangkaian pemilih fungsi. Contoh rangkaian logika 1 bit ditunjukkan pada gambar 10.



(a) (b)

Gambar 10. Rangkaian logika 1-bit: (a) rangkaian, (b) simbol

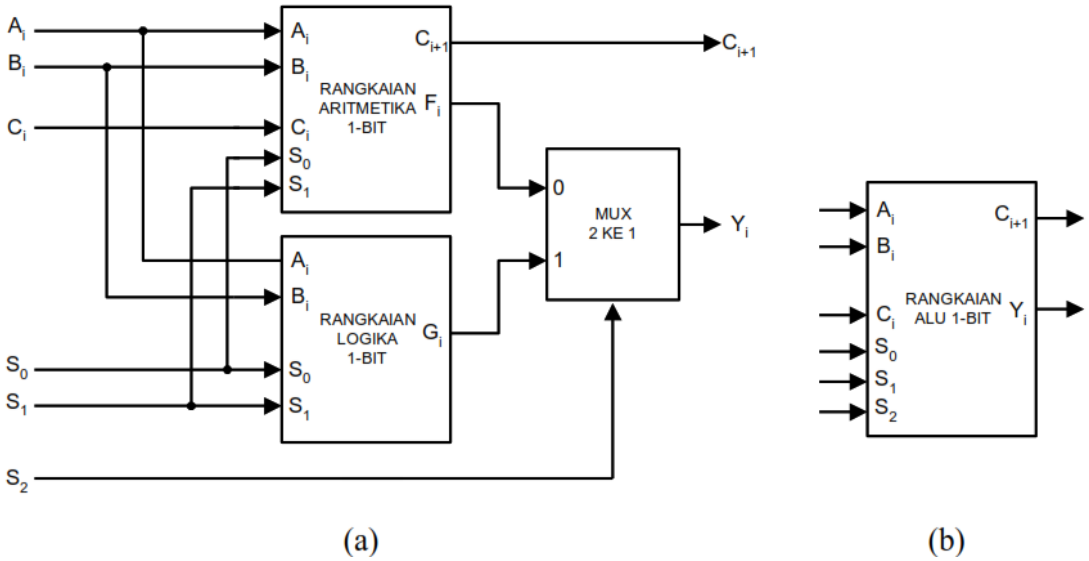
Dari rangkaian pada gambar 10, terlihat bahwa rangkaian logika tersebut dapat melaksanakan operasi logika sesuai dengan tabel 3 berikut ini.

Tabel 3. Tabel fungsi rangkaian logika

PEMILIH FUNGSI		OUTPUT	NAMA OPERASI
S_1	S_0	G	
0	0	G=A.B	AND
0	1	G=A+B	OR
1	0	G=A⊕B	XOR
1	1	G=not A	NOT

2. Rangkaian ALU 1-Bit

Rangkaian ALU 1-bit dibangun dengan menggabungkan rangkaian aritmetika 1-bit dan rangkaian logika 1-bit seperti ditunjukkan pada gambar 11.



Gambar 11. Rangkaian ALU 1-bit: (a) rangkaian, (b) simbol

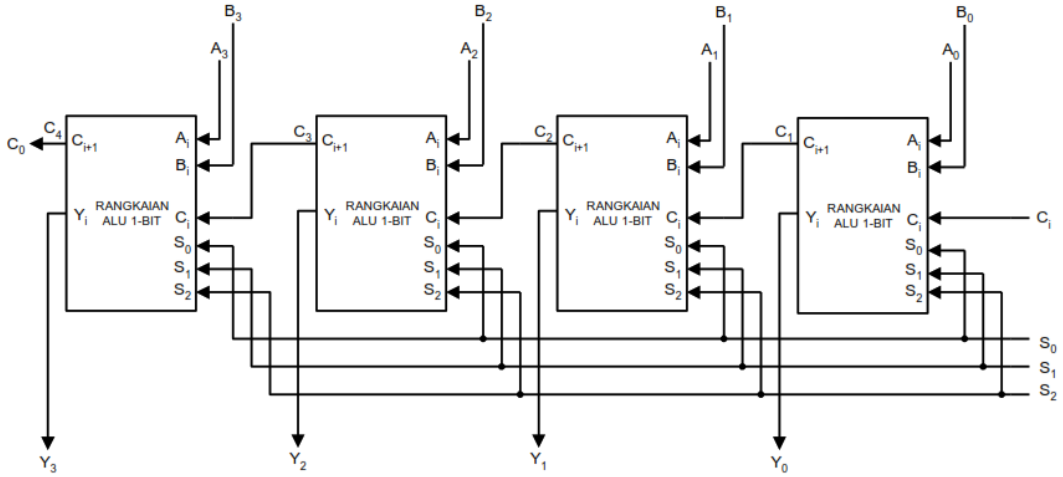
Karena rangkaian ALU 1-bit diperoleh dari penggabungan rangkaian aritmetika dan logika yang memiliki tabel fungsi seperti pada tabel 2 dan 3, maka kemampuan operasi rangkaian ini juga dicerminkan oleh gabungan dari kedua tabel tersebut. Berdasarkan hal tersebut, tabel fungsi rangkaian ALU ditunjukkan pada tabel 4 berikut ini.

Tabel 4. Tabel fungsi rangkaian ALU

PEMILIH FUNGSI				OUTPUT	NAMA OPERASI
S_2	S_1	S_0	C_i	Y	
0	0	0	0	Y=A	Transfer A
0	0	0	1	Y=A+1	Increment A
0	0	1	0	Y=A+not B	Penjumlahan A dengan not B
0	0	1	1	Y=A-B	Pengurangan
0	1	0	0	Y=A+B	Penjumlahan
0	1	0	1	Y=A+B+1	Penjumlahan dengan carry
0	1	1	0	Y=A-1	Decrement A
0	1	1	1	Y=A	Transfer A
1	0	0	X	Y=A.B	AND
1	0	1	X	Y=A+B	OR
1	1	0	X	Y=A⊕B	XOR
1	1	1	X	Y=not A	NOT

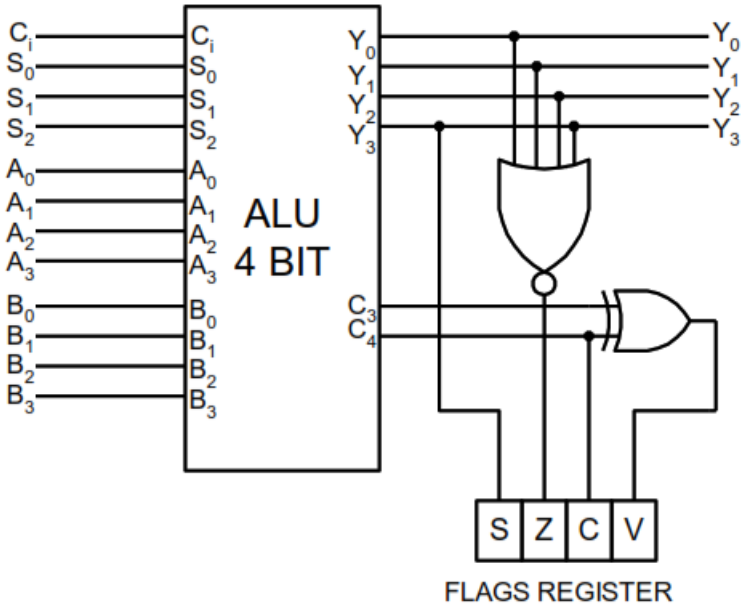
3. Rangkaian ALU n-Bit

Rangkaian ALU paralel dengan input dan output n-bit dibangun dengan menggunakan rangkaian ALU 1-bit yang disusun seperti rangkaian full adder paralel. Misal akan dibangun rangkaian ALU paralel 4-bit maka rangkaiannya adalah:



Gambar 12. Rangkaian ALU 4 bit

Untuk ALU yang dilengkapi dengan register bendera (flags register) ditunjukkan pada gambar 13.



Gambar 13. Rangkaian ALU 4 bit dengan register bendera

Fungsi register bendera adalah untuk menunjukkan hasil komputasi olehALU. Pada rangkaian gambar 13 terdapat register bendera 4 bit terdiri atas bit *sign* (S), bit *zero* (Z), bit *carry* (C), dan bit *overflow* (V). Bit S akan set (S=1) jika hasil perhitungan ALU negatif dan sebaliknya akan reset (S=0) jika hasil perhitungan ALUpositif. Bit Z akan set (Z=1) jika hasil pemrosesan ALU nol, dan Z akan reset (Z=0) jika hasilnya

tidak nol. Bit C set (C=1) jika perhitungan ALU menghasilkan *carry*dan reset (C=0) jika tak menghasilkan *carry*. Bit V akan set (V=1) jika hasil perhitungan ALU *overflow,* dan reset (V=0) jika tak *overflow*.

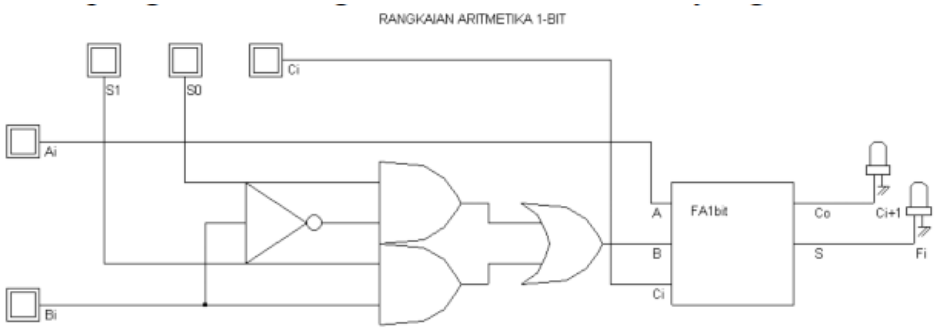
D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. Software Editor dan Simulator Logika DSCH2
3. Modul ARITMETIKA_1BIT.SCH, modul LOGIKA_1BIT.SCH, modul ALU_1BIT.SCH, dan modul ALU4BIT.SCH.

E. PROSEDUR PERCOBAAN

Rangkaian Aritmetika 1-Bit

1. Load file ARITMETIKA_1BIT.SCH sehingga Anda memperoleh tampilan seperti gambar 14.
2. Selidiki apakah rangkaian tersebut telah dapat melakukan operasi aritmetika sesuai dengan tabel 2? Untuk menyelidiki kemampuan operasi rangkaian ini gunakan tabel pengamatan rangkaian aritmetika 1-bit yang telah disediakan.

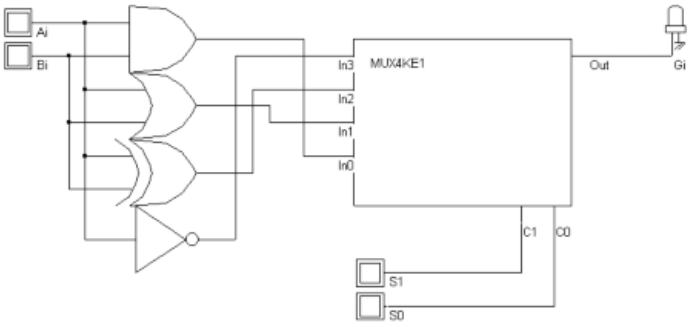


Gambar 14. Tampilan DSCH2 untuk rangkaian aritmetika 1-bit

3. Jika rangkaian telah dapat bekerja dengan baik, ubahlah rangkaian tersebut dalam bentuk simbol dengan nama ARITMETIKA_1BIT.SYM.

Rangkaian Logika 1-Bit

1. Load file LOGIKA_1BIT.SCH sehingga Anda memperoleh tampilan seperti gambar 15.

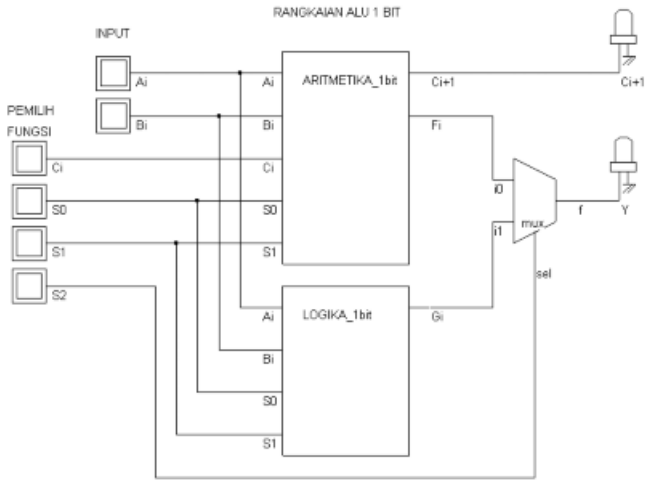


Gambar 15. Tampilan DSCH2 untuk rangkaian logika 1-bit

2. Selidiki apakah rangkaian tersebut telah dapat melakukan operasi logika sesuai dengan tabel 3? Untuk menyelidiki kemampuan operasi rangkaian ini gunakan tabel pengamatan rangkaian logika 1-bit yang telah disediakan.
3. Jika rangkaian telah dapat bekerja dengan baik, ubahlah rangkaian tersebut dalam bentuk simbol dengan nama LOGIKA_1BIT.SYM.

Rangkaian ALU 1-Bit

1. Dengan menggunakan simbol ARITMETIKA_1BIT.SYM dan LOGIKA_1BIT.SYM susunlah rangkaian ALU 1-bit seperti pada gambar 16. Simpan rangkaian tersebut ke dalam file ALU_1BIT.SCH.

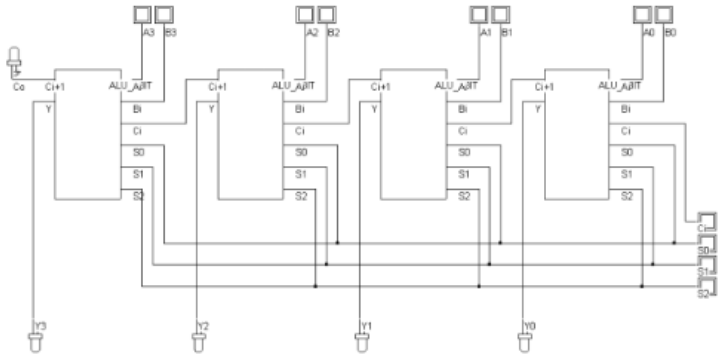


Gambar 16. Rangkaian ALU 1-bit hasil rancangan dengan DSCH2

2. Selidiki watak rangkaian ini apakah telah dapat melakukan operasi seperti pada tabel 4?
3. Jika rangkaian telah dapat bekerja dengan baik ubahlah rangkaian tersebut dalam bentuk simbol dengan nama ALU_1BIT.SYM.

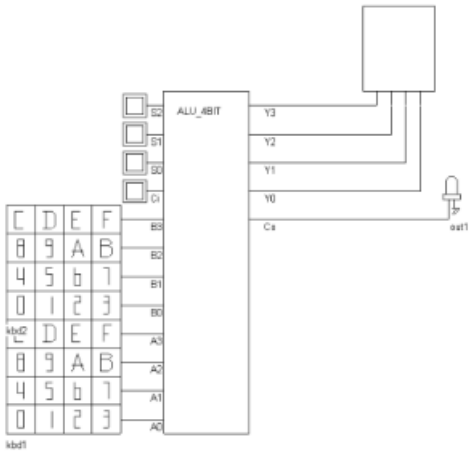
Rangkaian ALU 4-Bit

1. Dengan menggunakan simbol ALU_1BIT.SYM sebanyak 4 buah, susunlah rangkaian ALU 4-bit seperti pada gambar 17 dan simpan dalam file dengan nama ALU-4BIT.SCH.



Gambar 17. Rangkaian ALU 4-bit hasil rancangan dengan DSCH2

2. Selidiki watak rangkaian ini apakah telah dapat beroperasi seperti pada tabel 4?
3. Jika rangkaian telah dapat bekerja dengan baik ubahlah rangkaian tersebut dalam bentuk simbol dengan nama ALU_4BIT.SYM.
4. Mulailah dengan layar baru (new), load file ALU_4BIT.SYM dan susunlah rangkaian untuk menguji ALU 4-bit hasil rancangan seperti pada gambar 18, serta simpan rangkaian tersebut ke dalam file dengan nama ALU_4BIT_TES.SCH.

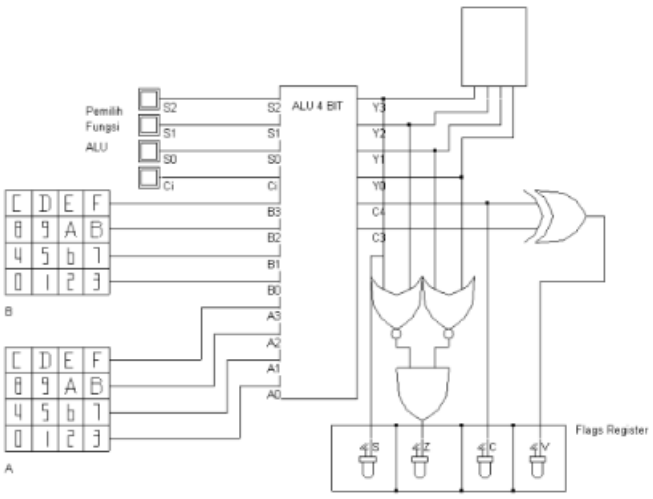


Gambar 18. Rangkaian pengujian ALU 4-bit dengan DSCH2

5. Selidiki watak rangkaian ini apakah telah dapat melakukan operasi seperti pada tabel 4?

Rangkaian ALU Dengan Register Bendera

1. Load file ALU_4BIT_FLAGS sehingga diperoleh tampilan seperti berikut ini:



Gambar 19. Rangkaian ALU dengan register bendera

2. Selidiki watak rangkaian ini apakah telah dapat melakukan operasi seperti pada tabel 4?

F. TUGAS AKHIR

1. Tuliskan persamaan output rangkaian aritmetika pada gambar 9. Dengan persamaan tersebut tunjukkan cara memperoleh tabel 2 yang merupakan tabel fungsi rangkaian aritmetika!
2. Apa fungsi register bendera pada ALU?
3. Pada percobaan Anda, kemukakan makna hasil pemrosesan ALU dengan melihat isi register bendera!

PERCOBAAN III
ORGANISASI BUS PROSESOR

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat:

1. Mengimplementasikan organisasi bus prosesor dengan beberapa register
2. Menjelaskan cara kerja ALU yang dilengkapi dengan akumulator.

B. TUGAS PENDAHULUAN

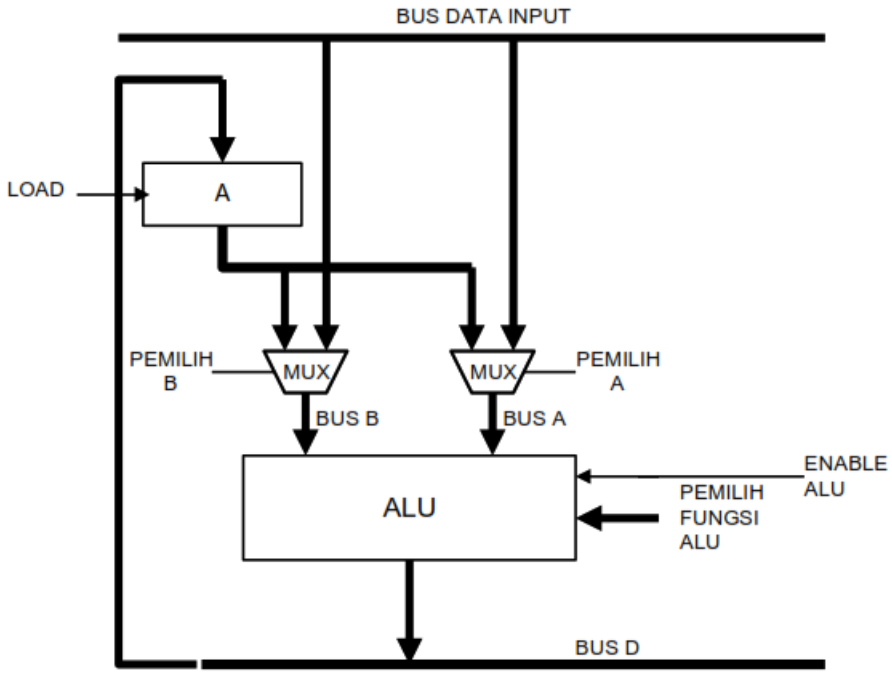
Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

1. Apa yang dimaksud dengan “bus prosesor” dalam konteks organisasi komputer?
2. Bagaimana bus prosesor berperan dalam mentransfer data dan instruksi antara komponen CPU dan memori utama?

C. TEORI

Dalam menjalankan fungsinya, ALU dilengkapi dengan register yang disebut akumulator. Fungsi akumulator adalah menampung data yang akan diproses dan sekaligus menampung data hasil pemrosesan oleh ALU. Dengan demikian akumulator berperan sebagai register sumber sebelum proses dilakukan dan sebagai register tujuan setelah proses dilakukan.

Interkoneksi antara ALU dengan akumulator-akumulator pendukungnya diimplementasikan dengan menggunakan transfer bus. Bentuk interkoneksi yang paling sederhana adalah antara ALU dengan sebuah akumulator yang ditunjukkan pada gambar 20 berikut ini.



Gambar 20. Organisasi bus prosesor dengan 1 akumulator

Dengan menggunakan perangkat keras pada gambar 20, penyelenggaraan suatu mikrooperasi dapat dijelaskan dengan mudah. Contoh: jelaskan urut-urutan proses dan pemberian sinyal kontrol pada mikrooperasi A□A+B. Dengan anggapan bahwa pemilih A dan pemilih B pada multiplexer A dan B memiliki tabel fungsi sebagai berikut.

Tabel 5. Tabel fungsi multiplexer pemilih isi bus A dan bus B

PEMILIH A	ISI BUS A
0	DATA INPUT
1	AKUMULATOR A

PEMILIH B	ISI BUS B
0	DATA INPUT
1	AKUMULATOR A

dan sinyal kontrol agar ALU aktif sebagai penjumlah adalah ENABLE ALU=1 dan $S_2S_1S_0C_i$=0100, maka urut-urutan pelaksanaan mikrooperasi A□A+B adalah sebagai berikut:

1. Mula-mula dianggap akumulator A dan bus data input telah tersisi dengan data. Pemilih MUX A diberi sinyal tinggi (Pemilih A=1) agar MUX A memindahkan isi akumulator A ke bus A.
2. Pemilih MUX B diberi sinyal rendah (Pemilih B=0) agar MUX B memindahkan isi bus data input ke bus B.
3. Pemilih fungsi ALU diatur dengan memberi nilai $S_2S_1S_0C_i$=0100 agar ALU dapat melaksanakan mikrooperasi penjumlahan.
4. ALU diaktifkan dengan memberikan ENABLE ALU=1 atau E=1. Dengan pengaktifan ALU ini, maka output ALU merupakan hasil penjumlahan dari isi bus A dan isi bus B. Oleh karena output ALU dihubungkan ke bus D, maka isi bus D sama dengan output ALU.
5. Untuk menyalin isi bus D ke akumulator A, LOAD pada akumulator diberi sinyal tinggi (LOAD=1) atau L=1. Dengan pemberian sinyal tersebut, isi akumulator A sama dengan isi bus D yang merupakan hasil pelaksanaan mikrooperasi penjumlahan.

Jadi, untuk melaksanakan mikrooperasi tersebut diperlukan sinyal kontrol $ABS_2S_1S_0C_iEL$=10010011. Sinyal tersebut dinamakan *control word*.

Interkoneksi yang lebih rumit ditemukan pada prosesor dengan ALU yang didukung oleh banyak register, seperti ALU dengan 3 buah register yakni A, B, C, dan sebuah bus data input. Mikrooperasi untuk mengisi BUS A dan BUS B padaALU dengan isi register-register yang tersedia dan input yang ada dapat dinyatakan:

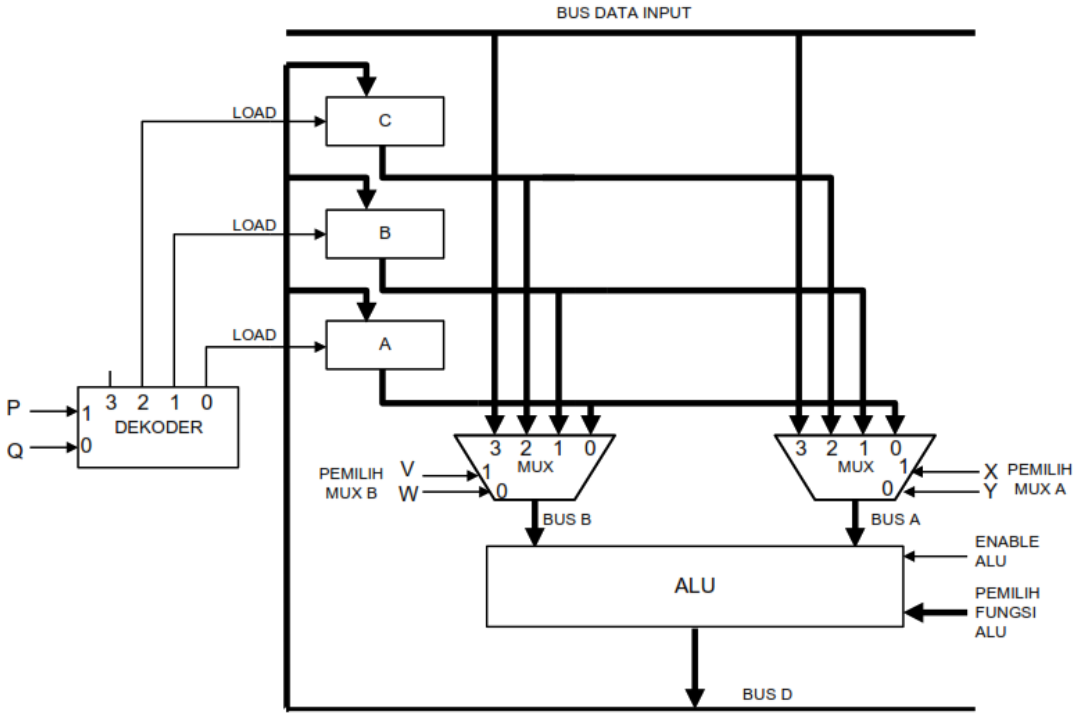
$\bar{X}\bar{Y}$: BUS A ← A
$\bar{X}Y$: BUS A ← A
$X\bar{Y}$: BUS A ← C
XY : BUS A ← INPUT

$\bar{V}\bar{W}$: BUS A ← A
$\bar{V}W$: BUS A ← A
$V\bar{W}$: BUS A ← C
VW : BUS A ← INPUT

Sedangkan mikrooperasi untuk memindahkan isi bus D ke setiap register yang diinginkan dinyatakan:

$\bar{P}\bar{Q}$: A ← A
$\bar{P}Q$: B ← B
$P\bar{Q}$: C ← C

Implementasi hardware dari ALU dengan 3 buah register dan sebuah bus data input dapat digambarkan sebagai berikut.

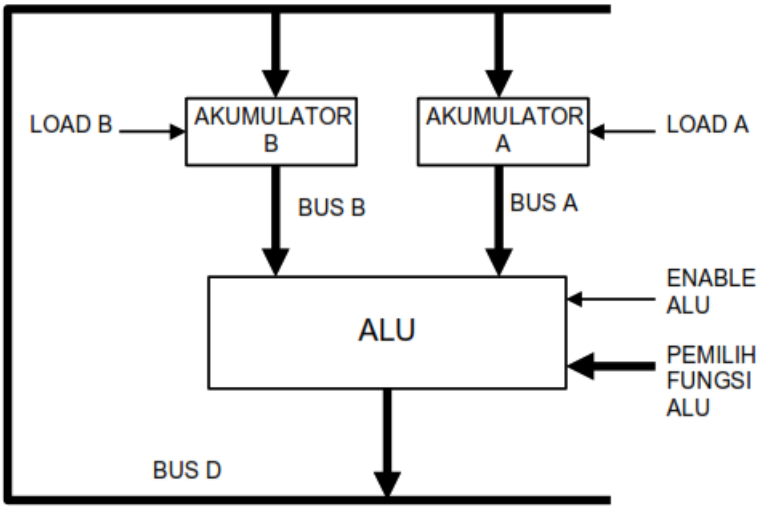


Gambar 21. Organisasi bus prosesor dengan 3 register

Dengan menggunakan gambar 21, dimisalkan akan diselenggarakan mikrooperasi A ← A+B, maka urut-urutan prosesnya adalah sebagai berikut:

1. Data pada register A ditempatkan pada bus A dengan memberi sinyal XY=00 pada pemilih MUX A.
2. Data pada register B ditempatkan pada bus B dengan memberi sinyal VW=01 pada pemilih MUX B.
3. ALU diatur agar mampu melaksanakan penjumlahan dengan memberi sinyal pemilih fungsi S2S1S0Ci=0100
4. ALU diaktifkan dengan memberi sinyal ENABLE ALU=1.
5. Pemilih tujuan diberi sinyal PQ=00 agar hasil pemrosesan ALU yang ada pada bus D disalin ke akumulator A.

Selain dapat diimplementasikan dengan menggunakan MUX dan decoder seperti pada gambar 20 dan 21, interkoneksi ALU dengan register-register pendukungnya juga dapat diimplementasikan dengan bentuk yang lebih sederhana. Gambar 22 berikut ini adalah contoh implementasi interkoneksi ALU dengan dua buah akumulator yang tidak melibatkan MUX dan decoder.



Gambar 22. ALU dengan 2 akumulator

Jika suatu mikrooperasi misalnya A←A+B akan diselenggarkan menggunakan perangkat keras pada gambar 22, maka urut-urutan prosesnya adalah sebagai berikut:

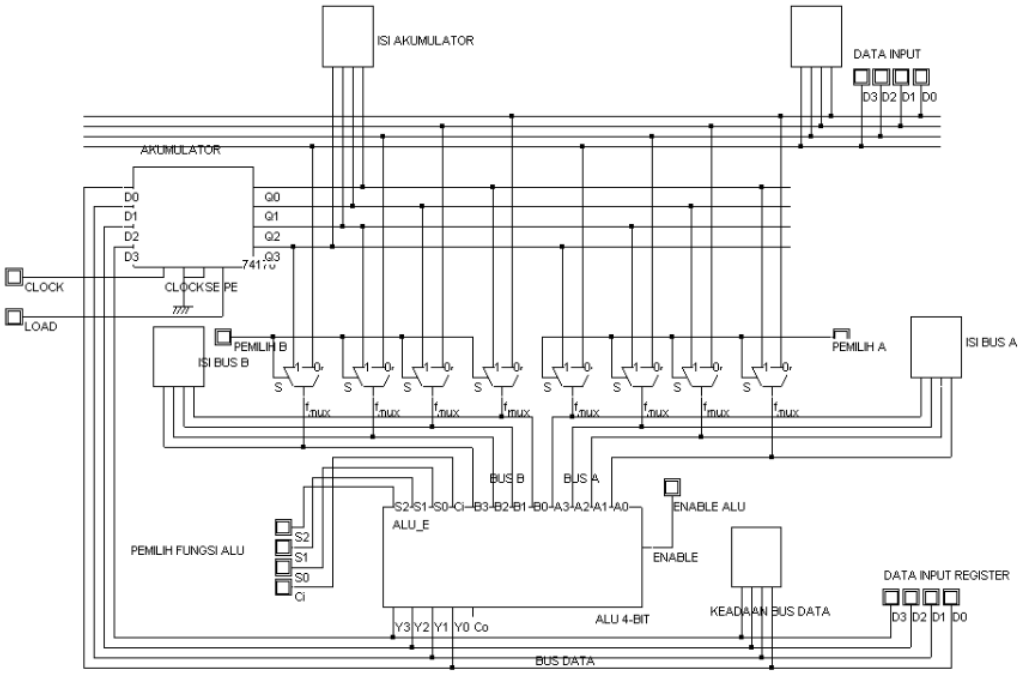
1. Bus D diisi data augend (bilangan yang akan dijumlahkan) terlebih dahulu. Selanjutnya dilakukan penyalinan data dari bus D ke akumulator A dengan memberi LOAD A=1.
2. Bus D diisi data addend (bilangan penjumlah). Kemudian dilakukan penyalinan data dari bus D ke akumulator B dengan memberi LOAD B=1.
3. Pada tahap ini bus A telah terisi oleh akumulator A dan bus B oleh akumulator B.
4. Pemilih fungsi ALU diatur dengan memberi sinyal S2S1S0Ci=0100 sehingga ALU sebagai penjumlah. Selanjutnya, diberikan sinyal ENABLE ALU=1 agar hasil penjumlahan disalin ke bus D.
5. Untuk menyalin hasil pemrosesan ALU yang ada pada bus D ke akumulator A, diberikan sinyal LOAD A=1.

D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul PROSESOR0.SCH, dan modul PROSESOR1.SCH

E. PROSEDUR PERCOBAAN

1. Load file PROSESOR0.SCH sehingga diperoleh tampilan layar monitor seperti pada gambar 23.
2. Isilah akumulator dengan data 5. Lakukan dengan mengisi terlebih dahulu bus data dengan data 5 atau 0101. Selanjutnya, berikan LOAD=1 dan sebuah *clock* sehingga akumulator terisi data 5.
3. Isilah bus data input dengan data 2 atau 0010.
4. Berikan PEMILIH A=1 sehingga isi akumulator disalin ke bus A.
5. Berikan PEMILIH B=0 sehingga isi bus data input disalin ke bus B. Pada tahap ini terlihat bahwa isi bus A adalah 5 dan isi bus B adalah 2.
6. Berikan sinyal $S_2S_1S_0C_i$=0100 pada ALU sehingga ALU menjumlah isi bus A dan isi bus B.
7. Berikan sinyal ENABLE ALU=1 sehingga isi bus D merupakan penjumlahan isi bus A dan bus B.
8. Untuk menyalin isi bus D ke akumulator berikan sinyal LOAD=1 dan sebuah sinyal *clock.*



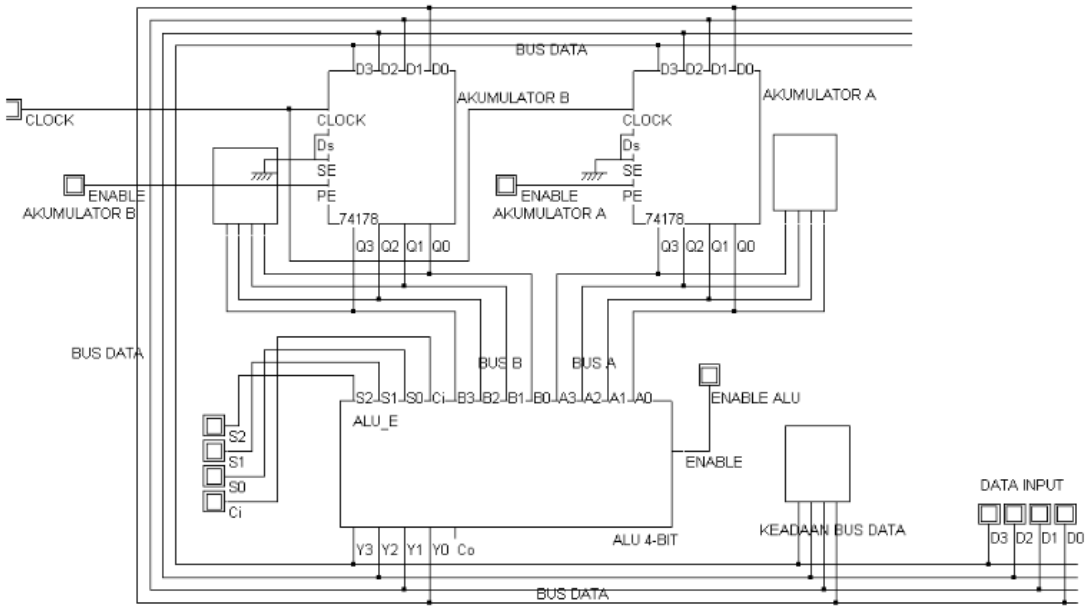
Gambar 23. Tampilan DSCH2 untuk rangkaian ALU dengan 1 akumukator

9. Coba ulangi percobaan untuk menyelenggarakan mikrooperasi yang lain seperti transfer, pengurangan, increment, decrement, dan penjumlahan dengan carry!

ALU dengan 2 Akumulator

1. Load file PROSESOR1.SCH sehingga diperoleh tampilan seperti pada gambar 24.
2. Isilah bus data dengan data 3.

3. Salin isi bus data ke akumulator A dengan memberikan LOAD A=1 dan sebuah sinyal clock. Setelah akumulator A berisi data 3, matikan load denganmemberikan LOAD A=0.
4. Isi bus data dengan data 5.
5. Salin isi bus data ke akumulator B dengan memberikan LOAD B=1 dan sebuah sinyal clock. Setelah akumulator B berisi data 5, matikan load denganmemberikan LOAD B=0.
6. Berikan sinyal $S_2S_1S_0C_i$=0100 pada ALU sehingga ALU menjumlah isi bus A dan isi bus B.
7. Berikan sinyal ENABLE ALU=1 sehingga isi bus D merupakan penjumlahan isi bus A dan bus B.
8. Untuk menyalin isi bus D ke akumulator A berikan sinyal LOAD A=1 dan sebuah sinyal *clock.*
9. Coba ulangi percobaan untuk menyelenggarakan mikrooperasi yang lain seperti transfer, pengurangan, increment, decrement, dan penjumlahan dengan carry!



Gambar 24. Tampilan DSCH2 untuk rangkaian ALU dengan 2 akumulator

F. TUGAS AKHIR

1. Susun rangkaian ALU dengan 7 register dan sebuah bus data input! Jelaskan urut-urutan proses jika diselenggarakan mikrooperasi A←A+B, B←A-B, A←A, D←A-B, dan B←C+D!
2. Apa yang dimaksud *control word*! Tulislah *control word* yang diperlukan untuk menyelenggarakan mikrooperasi menjumlahkan bilangan 3 dengan 5 dan hasilnya disimpan di akumulator A seperti yang sudah Anda coba pada percobaan kedua? Tuliskan pula *control word* yang diperlukan pada beberapa mikrooperasi tugas nomor 1!

PERCOBAAN IV
SISTEM MEMORI

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat:

1. Merancang unit memori ROM dan RAM sederhana
2. Mensimulasikan kerja prosesor dalam mengakses unit memori.

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

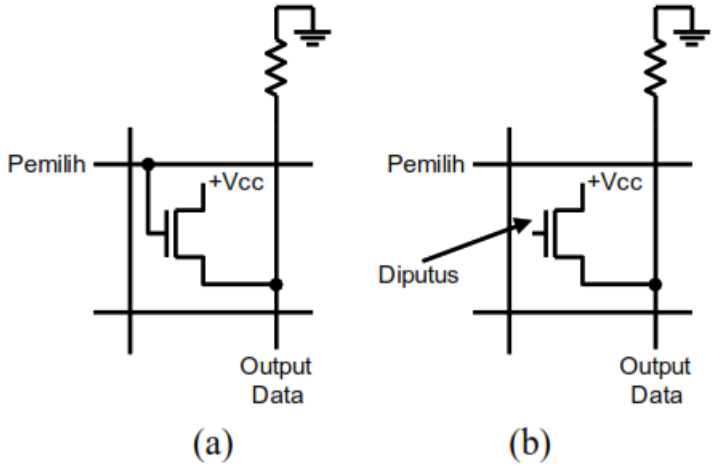
1. Apa yang dimaksud dengan kapasitas memori, dan mengapa kapasitas ini penting dalam desain komputer?
2. Apa yang dimaksud dengan kecepatan akses memori, dan mengapa itu menjadi faktor kunci dalam kinerja komputer?

C. TEORI

Unit memori merupakan salah satu bagian dalam sistem komputer yang berfungsi menyimpan data dan program. Secara umum memori dibagi menjadi dua bagian yakni ROM singkatan dari *read only memory* dan RAM singkatan dari *random access memory*. ROM hanya dapat dibaca saja dan tidak dapat ditulisi serta sifat penyimpanannya permanen. Isi ROM tidak akan hilang walaupun catu dayanya ditiadakan. Sedangkan RAM kecuali dapat ditulisi juga dapat dibaca, tetapi sifat penyimpanannya sementara, dalam hal ini isi RAM akan hilang jika catu dayanya ditiadakan.

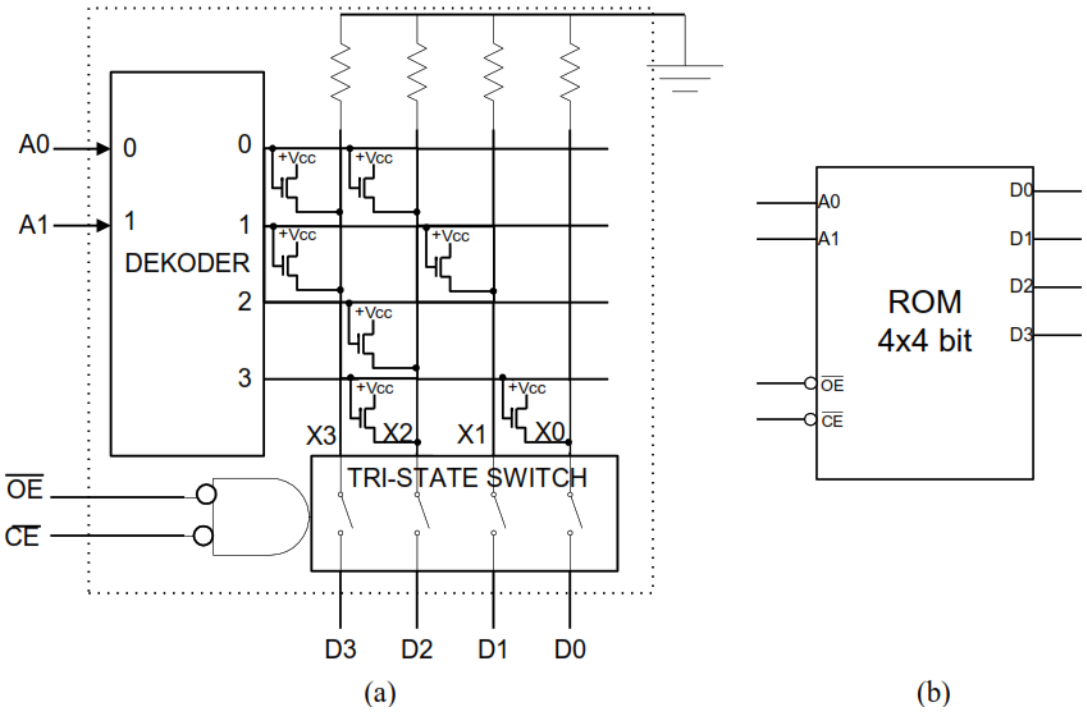
Jenis ROM terbagi menjadi MPROM (*mask programmed* ROM) atau PROM, EPROM (*erasable programmable* ROM), dan EEPROM (*electrically* EPROM).Jenis pertama yakni PROM merupakan ROM yang diprogram oleh pabrik sesuai pesanan pengguna dan dan hanya sekali dapat diprogram. EPROM merupakan ROM yang dapat dihapus isinya dan diprogram kembali. Salah satu jenis EPROM adalah UV-EPROM (ultra violet EPROM) yang dapat dihapus isinya dengan menyinari sel- selnya dengan sinar ultra violet beberapa menit. Untuk memrogram EEPROM digunakan EPROM programmer. Sedangkan EEPROM merupakan ROM yang dapat dihapus dan diprogram isinya secara elektris.

Sel PROM berupa sebuah transistor yang berfungsi menyimpan data 1 dan 0 seperti ditunjukkan pada gambar 25



Gambar 25. Sel ROM: (a) penyimpan data 1, (b) penyimpan data 0

Cara kerja sel ROM sangat sederhana yakni ketika sinyal pemilih diberikan, untuk gambar 25 (a) akan menyebabkan transistor ON sehingga jalur output data yang semula bernilai 0 menjadi tinggi atau bernilai 1. Sedangkan pada gambar (b), pemberian sinyal pemilih tidak berpengaruh karena *gate* transistor tidak terhubung ke pemilih sehingga transistor tetap dalam keadaan OFF. Akibatnya pada jalur output tetap memberikan nilai rendah atau 0.



Gambar 26. ROM 4X4 bit: (a) Organisasi, dan (b) simbol

Diumpamakan akan dibangun ROM ukuran 4 x 4-bit dengan menggunakan sel transitor dan akan digunakan untuk menyimpan data seperti tabel berikut ini.

Tabel 6. Data yang akan disimpan pada ROM ukuran 4X4 bit

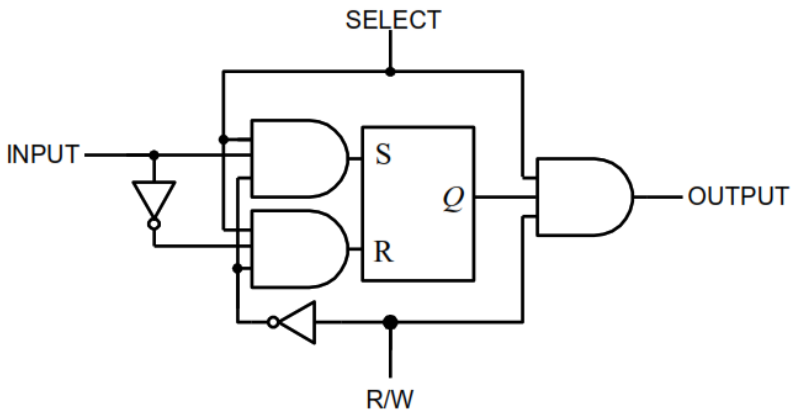
LOKASI/ ALAMAT	DATA DALAM BINER				DATA DALAM HEKSADESIMAL
	D_3	D_2	D_1	D_0	
0	1	1	0	0	C
1	1	0	1	0	A
2	0	1	0	0	4
3	0	1	0	1	5

Untuk sel-sel yang menyimpan data 1 transistornya digambarkan, tetapi sel- sel yang menyimpan data 0 transistornya tidak digambarkan agar tidak kelihatan rumit.

Sel-sel yang telah disusun selanjutnya dilengkapi dengan dekoder sebagai pemilih lokasi/alamat, dan rangkaian *tri-state switch* sebagai pengontrol ROM. Untuk ROM ukuran 4X4-bit memiliki pin/terminal data sebanyak 4-bit yakni D_3, D_2, D_1, dan D_0, pemilih sebanyak 2 bit yakni A_1 dan A_0, dan untuk selanjutnya pemilih ini dinamakan pin/terminal alamat, serta pin kontrol CE (*chip enable*) dan OE (*output enable*). Pin CE berfungsi untuk mengaktifkan ROM, dan karena jenisnya *active-low*, maka ROM akan aktif jika pin CE=0. Pin OE berfungsi untuk menyediakan data pada saluran outputnya, dan karena jenisnya *active-low*, ROM dapat dibaca jika OE=0.

Cara mikroprosesor membaca ROM dapat dijelaskan sebagai berikut.Misalnya akan dibaca alamat 2, maka mikroprosesor mengirim terlebih dahulu lewat bus alamat sinyal alamat bernilai 2 yakni A_1A_0=10. Selanjutnya, lewat bus kontrol mikroprosesor mengirim sinyal CE=0 dan OE=0. Dengan cara seperti itu maka isi alamat 2 yakni data 2 atau 0100 akan ditempatkan pada bus data.

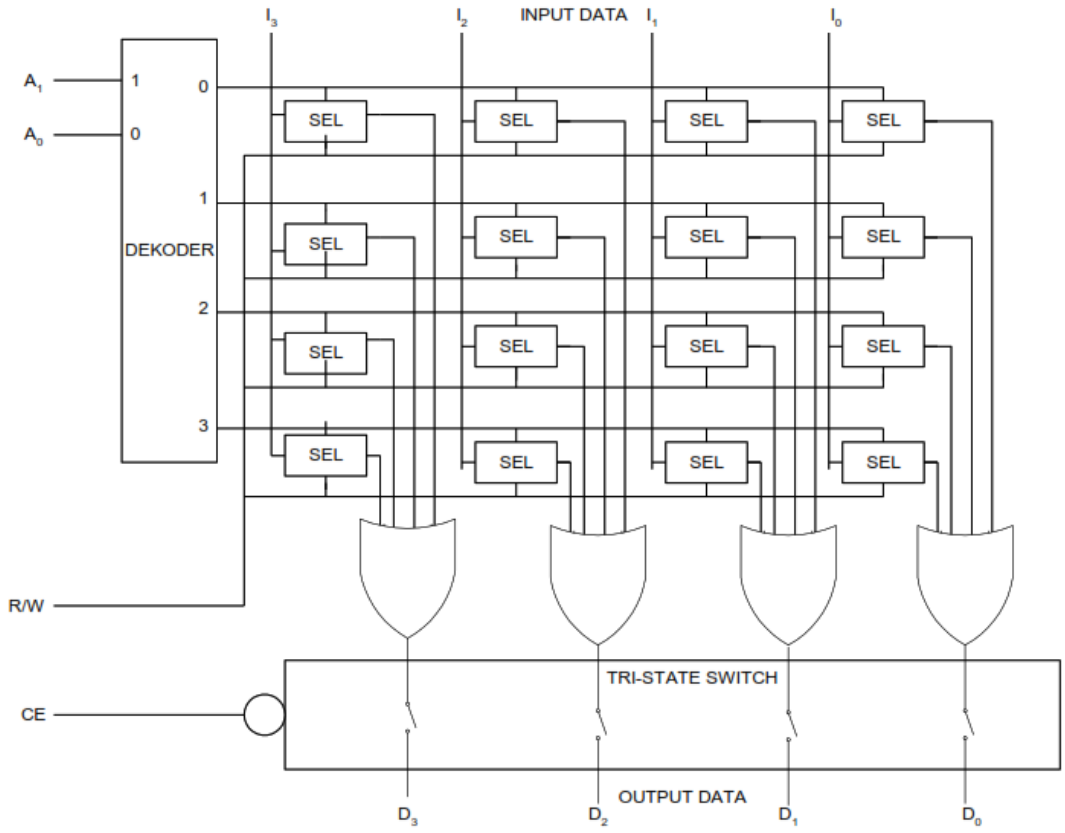
RAM dapat diklasifikasikan menjadi SRAM (*static* RAM) atau RAM statis dan DRAM (*dynamic* RAM) atau RAM dinamis. Perbedaan yang paling mendasar dari kedua jenis RAM tersebut adalah dari segi sel penyusunnya. SRAM menggunakan sel yang terbuat dari flip-flop sehingga stabilitas data lebih baik dibandingkan RAM dinamis yang menggunakan sel dari transistor dan kapasitor, sehingga data pada DRAM harus selalu di *refresh*. Karena sel-selnya terbuat dari flip-flop maka untuk kapasitas yang sama ukuran SRAM lebih besar dibandingkan ukuranfisik DRAM. Sel SRAM dapat digambarkan seperti pada gambar 27.



Gambar 27. Sel SRAM

Untuk dapat mengakses sel tersebut diperlukan sinyal SELECT=1 agar memungkinkan semua gerbang AND *enable*. Pada mode operasi tulis/*write*, R/W harus bernilai rendah (R/W=0). Jika R/W=0 maka output akan *inhibit* (*disable*), sedangkan inputnya akan *enable* sehingga data dapat disimpan pada flip-flop. Pada operasi baca/*read*, R/W harus tinggi (R/W=1). Jika R/W=1 maka kedua gerbangAND pada input sel *inhibit* sehingga operasi tulis tak dapat dilakukan, dan sebaliknya gerbang AND pada output akan *enable* sehingga output sel dapat disalurkan ke output.

SRAM disusun dari sel-sel seperti pada gambar 27, dan dilengkapi dengan rangkaian pemilih dan pengontrol sehingga membentuk organisasi RAM seperti ditunjukkan gambar 28.



Gambar 28. Organisasi SRAM ukuran 4x4 bit

Pada RAM ukuran 4x4 bit memiliki 4 bit saluran input dan 4bit saluran output, 2 buah pin alamat, dan pin kontrol CE serta R/W. Dalam hal ini pin CE jenisnya *active-low* sehingga untuk mengaktifkan RAM nilai CE harus rendah (CE=0). Mode operasi tulis/*write* diatur dengan memberikan R/W=0 dan mode baca/*read* diatur dengan memberi R/W=1.

D. ALAT-ALAT PERCOBAAN

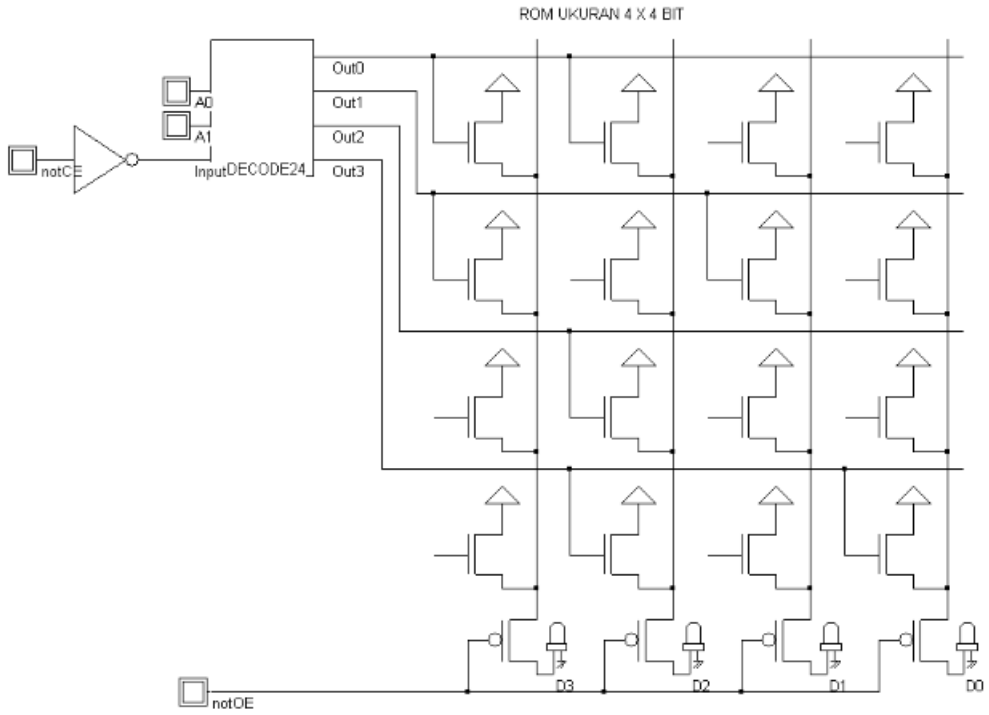
1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2

3. Modul ROM4X4.SCH, modul ROM4X4TES.SCH, dan modul RAM8X4_TES.SCH

E. PROSEDUR PERCOBAAN

Merancang ROM Ukuran 4X4 Bit

1. Load modul ROM4X4.SCH sehingga diperoleh tampilan seperti pada gambar 29.



Gambar 29. Tampilan DSCH2 untuk rangkaian ROM 4X4 bit

2. Selidiki isi ROM tersebut dan tuliskan isi dari masing-masing alamat/lokasi!
3. Dengan menggunakan gambar 29, rancanglah ROM agar dapat menyimpan data sebagai berikut:

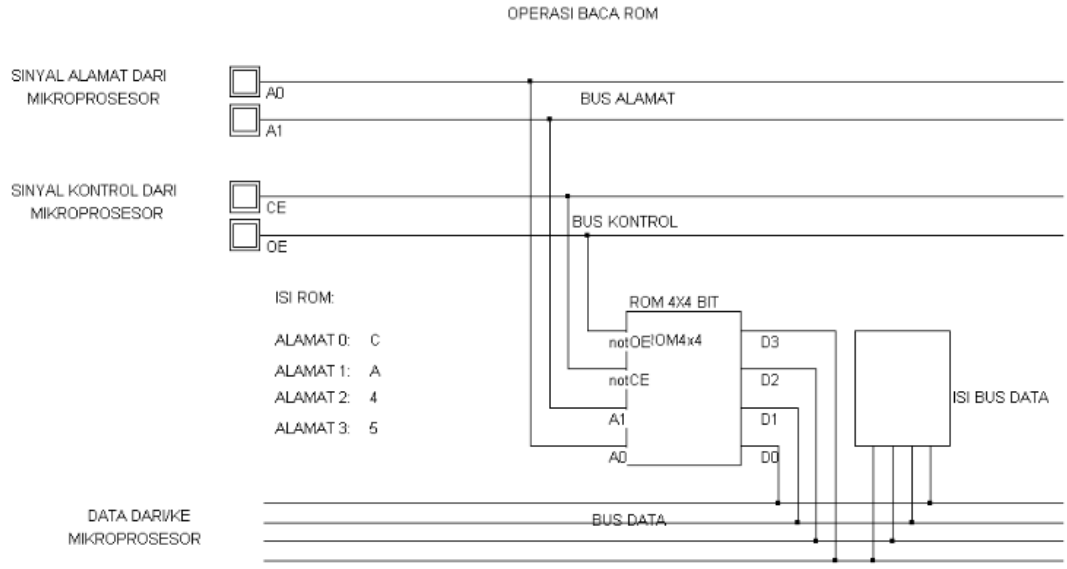
Tabel 7. Data yang akan disimpan pada ROM untuk tugas rancangan

LOKASI/ ALAMAT	DATA DALAM BINER				DATA DALAM HEKSADESIMAL
	D_3	D_2	D_1	D_0	
0	0	0	1	1	3
1	1	0	0	0	8
2	1	0	1	1	B
3	1	1	1	0	E

4. Lakukan pengujian terhadap hasil rancangan anda apakah ROM sudah berisi data seperti pada tabel 7?

Operasi Baca Pada ROM

1. Load modul ROM4X4TES.SCH sehingga diperoleh tampilan layar monitor seperti pada gambar 30.
2. Pada modul tersebut, diumpamakan ROM dihubungkan dengan mikroprosesor melalui sistem interkoneksi bus. Pin alamat ROM dihubungkan dengan bus alamat mikroprosesor, pin data ROM dengan bus data mikroprosesor, dan pin kontrol ROM dengan bus kontrol mikroprosesor. ROM yang digunakan berukuran 4X4 bit dan telah diprogram sehingga keempat alamat atau lokasinya telah terisi dengan data.

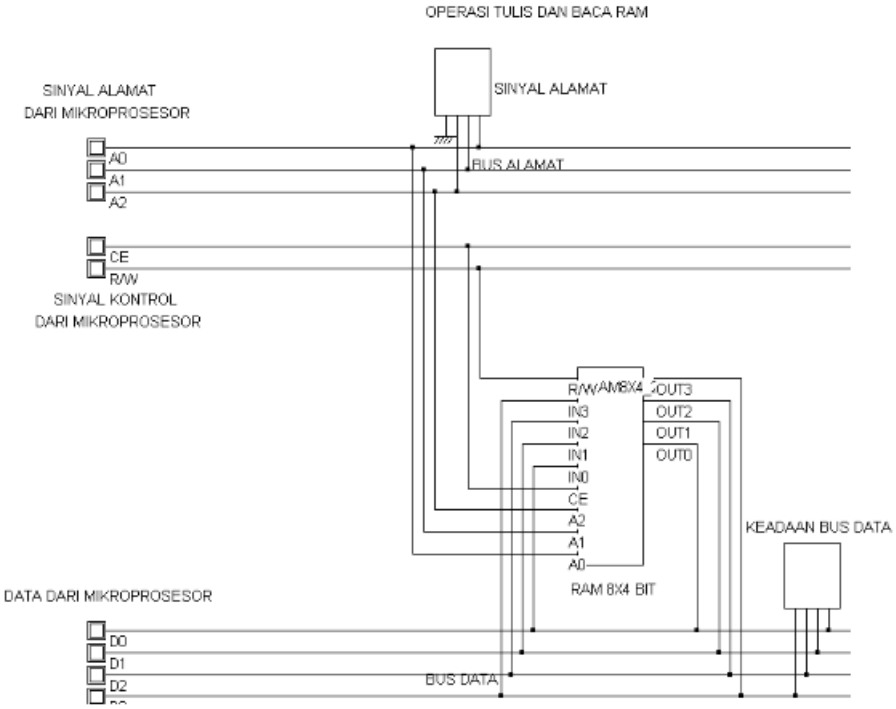


Gambar 30. Interkoneksi ROM dengan bus mikroprosesor untuk operasi baca

3. Lakukan pembacaan setiap alamat dari ROM dengan memberi sinyal alamat terlebih dahulu pada bus alamat, dilanjutkan dengan memberi sinyal kontrol CE=0 dan OE=0.
4. Apa yang tejadi pada setiap pembacaan alamat jika sinyal kontrol CE=1?

Operasi Tulis/Write Pada RAM

1. Load modul RAM8X4_TES.SCH sehingga diperoleh tampilan seperti pada gambar 31.



Gambar 31. Interkoneksi RAM dengan bus mikroprosesor untuk operasi baca

2. Diumpamakan ke dalam alamat-alamat RAM akan diisikan data sebagai berikut:

Tabel 8. Data yang akan disikan ke dalam alamat RAM

ALAMAT	DATA BINER				DATA HEKSADESIMAL
	D_3	D_2	D_1	D_0	
0	1	0	1	0	A
1	1	0	1	1	B
2	1	1	0	0	C
3	1	1	0	1	D
4	1	1	1	0	E
5	1	1	1	1	F
6	0	0	0	1	1
7	0	0	1	0	2

3. Berikan terlebih dahulu sinyal R/W=1 agar RAM tidak aktif pada mode tulis. Tempatkan sinyal alamat yang akan ditulisi pada bus alamat, misalnya $A_2A_1A_0$=000, tempatkan data pada bus data misalnya $D_3D_2D_1D_0$=1010 atau A. Berikan sinyal tulis R/W=0 sesaat kemudian kembalikan R/W=1. Dengan cara tersebut seolah-olah mikroprosesor telah menyimpan data 1010 biner atau data A heksadesimal ke alamat 0 pada RAM.
4. Ulangi langkah 3 untuk menyimpan data pada alamat 1, 2, 3, 4, 5, 6, dan 7 dengan data seperti pada tabel 8.

Operasi Baca/Read Pada RAM

1. Setelah Anda melakukan penyimpanan data pada RAM, sekarang lakukan pembacaan pada isi alamat-alamat dari RAM tersebut.
2. Pastikan sinyal kontrol masih dalam keadaan CE=0 dan R/W=1.
3. Berikan sinyal alamat 0 atau $A_2A_1A_0$=000, dan amati isi bus data! Isi bus data menunjukkan isi alamat yang dibaca. Catat isi alamat 0!
4. Ulangi pembacaan untuk alamat 1, 2, 3, 4, 5, 6, dan alamat 7!

F. TUGAS AKHIR

1. Rancanglah ROM ukuran 8X8 bit untuk menyimpan data heksadesimal A0 pada alamat 0,B1 pada alamat 1, C2 pada alamat 2, D3 pada alamat 3, 4E pada alamat 4, 5F padaalamat 5, 65 pada alamat 6, dan 74 pada alamat 7.
2. Jelaskan cara mikroprosesor membaca isi alamat 5 pada rancangan ROM soal nomor 1!
3. Rancanglah SRAM ukuran 8X8 bit. Jelaskan cara mikroprosesor menulisi alamat 3 dari RAM tersebut dengan data F5! Jika alamat 7 dari RAM itu dianggap telah berisi data FF, jelaskan cara mikroprosesor membaca alamat 7!

PERCOBAAN V
VSM (VERY SIMPLE MICROPROCESSOR)
BAGIAN I: UNIT AKUMULATOR

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat membangun dua macam unit akumulator pada mesin VSM.

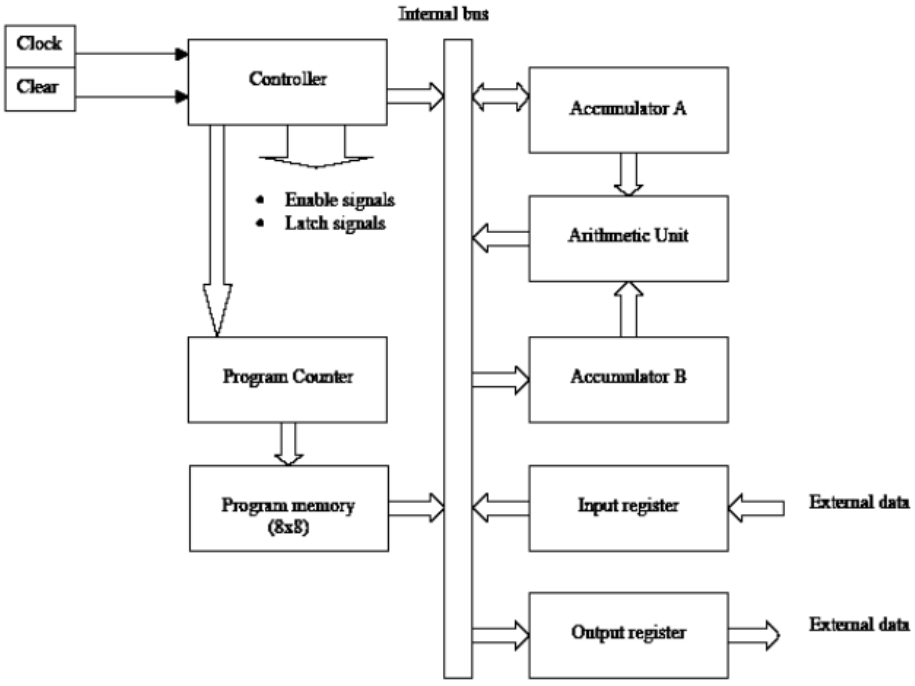
B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

1. Apa fungsi utama Unit Akumulator dalam Very Simple Microprocessor (VSM)?
2. Bagaimana Unit Akumulator digunakan dalam operasi aritmetika seperti penjumlahan dan pengurangan?

C. TEORI

VSM merupakan bentuk paling sederhana dari arsitektur mikroprosesor. Mesin ini digunakan untuk memperkenalkan konsep dasar arsitektur komputer yang dikembangkan oleh Sicard (2005) dan merupakan versi turunan dari mesin yang diusulkan oleh Malvino (1993) yakni *Simple-as-Possible Processor*. Dengan menggunakan VSM dapat dipelajari cara kerja sistem komputer dengan mudah. Arsitekturnya yang sangat primitif, seperti ditunjukkan pada gambar 32, menjadikan mesin ini nampak tidak terlalu rumit.



Gambar 32. Arsitektur mesin VSM

Program counter merupakan pencacah 4-bit yang melakukan pencacahan dari 0000 sampai dengan 1111. Outputnya merupakan kode alamat instruksi yang akan diambil dari lokasi memori. Keadaan awal dari *program counter* adalah 0000 yang berarti mikroprosesor mulai bekerja dengan melaksanakan instruksi yang tersimpan pada alamat 0000.

Program memory merupakan memori yang menyimpan program. Masing-masing baris program mempunyai format 8-bit yakni 4-bit pada bagian atas sebagai kode instruksi (*opcode*) dan 4-bit bagian bawah sebagai data yang dioperasikan (operand).

Accumulator A merupakan register 4-bit yang berfungsi menyimpan secara sementara hasil yang diperoleh dari komputasi mikroprosesor. Register ini dilengkapi dengan kontrol *enable*A. Jika *enableA* diberi sinyal tinggi, data yang ada di dalamnya akan ditempatkan pada bus internal.

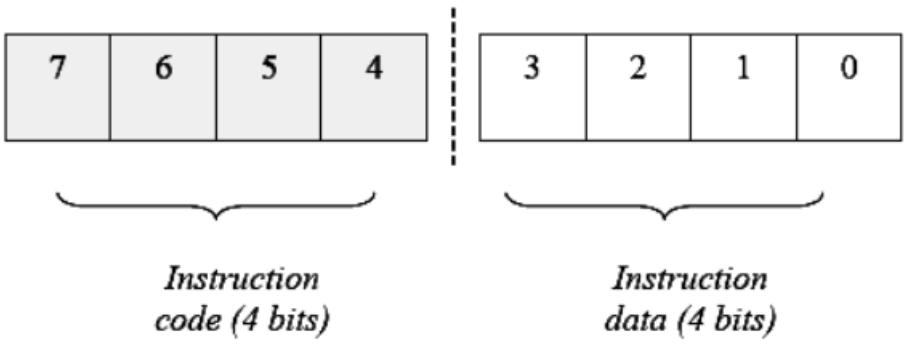
Accumulator B merupakan register 4-bit yang berfungsi menyimpan data yang akan ditambahkan atau dikurangkan pada operasi penambahan atau pengurangan.

Arithmetic unit merupakan rangkaian aritmetika dengan kemampuan melaksanakan operasi penjumlahan (S=A+B) dan pengurangan (S=A+$\bar{B}$+1).

Input register merupakan register input yang berfungsi menyimpan data dari luar sistem yang akan dimasukkan ke dalam mikroprosessor. Sedangkan *output register* merupakan register yang berfungsi menyimpan data yang akan ditransfer dari bus internal ke luar sistem.

1. Memori Program

Memori pada mesin ini berukuran 8x8 bit yang digunakan untuk menyimpan instruksi yang akan dilaksanakan. Masing-masing instruksi dikode dalam format 8 bit. Empat bit pada bagian orde tinggi merupakan instruksi dan 4 bit pada bagian orde rendah merupakan operand atau data. Program yang disimpan dalam memori adalah instruksi-instruksi untuk melaksanakan proses: mengisi *accumulator A* dengan data 2 kemudian menambahkannya dengan data 1 dan menempatkan hasilnya pada register output.



Gambar 33. Format instruksi pada VSM

Program yang disimpan ke dalam memori terdiri atas 4 instruksi ditunjukkan pada tabel 10.

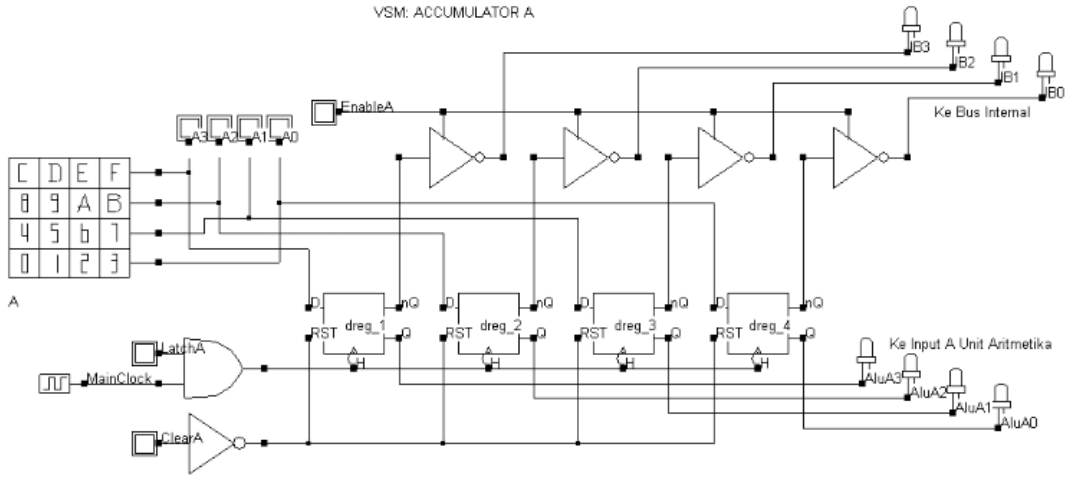
Tabel 10. Daftar instruksi yang disimpan ke dalam memori

Mnemonic	Opcode | Operand Dalam Biner	Opcode | Operand Dalam Heksadesimal
LDA 2	0101 0010	52
ADD 1	0001 0001	11
OUT	0100 0000	40
NOP	0000 0000	00

2. Rancangan Blok Dasar Mesin VSM (Bagian 1)

Accumulator A

Accumulator A dibangun dari 4 buah flip-flop D jenis *negative-edge triggered*, sehingga akumulator akan merespons inputnya jika clock berubah dari tinggi ke rendah. Pada operasi ADD dan SUB, output akumulator terhubung secara langsung ke unit aritmetika melalui Alu3, Alu2, Alu1, dan Alu0. *Accumulator* A dilengkapi dengan pengontrol *EnableA* jenis *active-high.* Dalam hal ini jika *EnableA*=1, isi akumulator akan ditransfer ke bus internal. Untuk membangun fasilitas *enable* digunakan empat buah *tri-state inverter*.

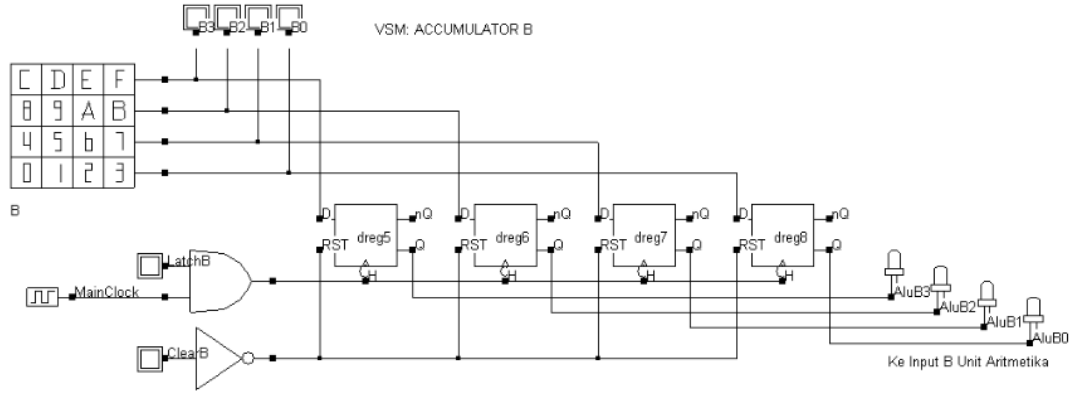


Gambar 34. Struktur *accumulator* A

Selain itu, *accumulator A* dilengkapi pula dengan pengontrol *LatchA* jenis *active-high*. Jika *LatchA*=1, data dari input akan ditransfer ke sel-sel akumulator. Untuk mengosongkan isi akumulator, disediakan input *ClearA* jenis *active-low*. Jika *ClearA=0* maka data dalam sel-sel akumulator akan dikosongkan. Agar akumulator dapat bekerja secara normal, *ClearA* harus selalu tinggi (*ClearA*=1).

Accumulator B

Seperti halnya *accumulator A, accumulator B* juga dibangun dari empat buah flip-flop D. Pada operasi ADD atau SUB, output register ini langsung terhubung ke input B unit aritmetika melalui AluB3, AluB2, AluB1, dan AluB0. Pengontrol *LatchB* jenis *active-high* disediakan untuk mentransfer data dari input ke sel-sel akumulator. Sedangkan untuk menghapus isi akumulator disediakan pengontrol *ClearB* jenis *active-low*, sehingga pengontrol ini perlu dipertahankan tinggi agar akumulator bekerja secara normal. Rangkaian *accumulator B* dalam versi editor DSCH2 ditunjukkan pada gambar 35.



Gambar 35. Struktur *accumulator* B

D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul VSM_AccumulatorA.SCH, Modul VSM_AccumulatorB.SCH,

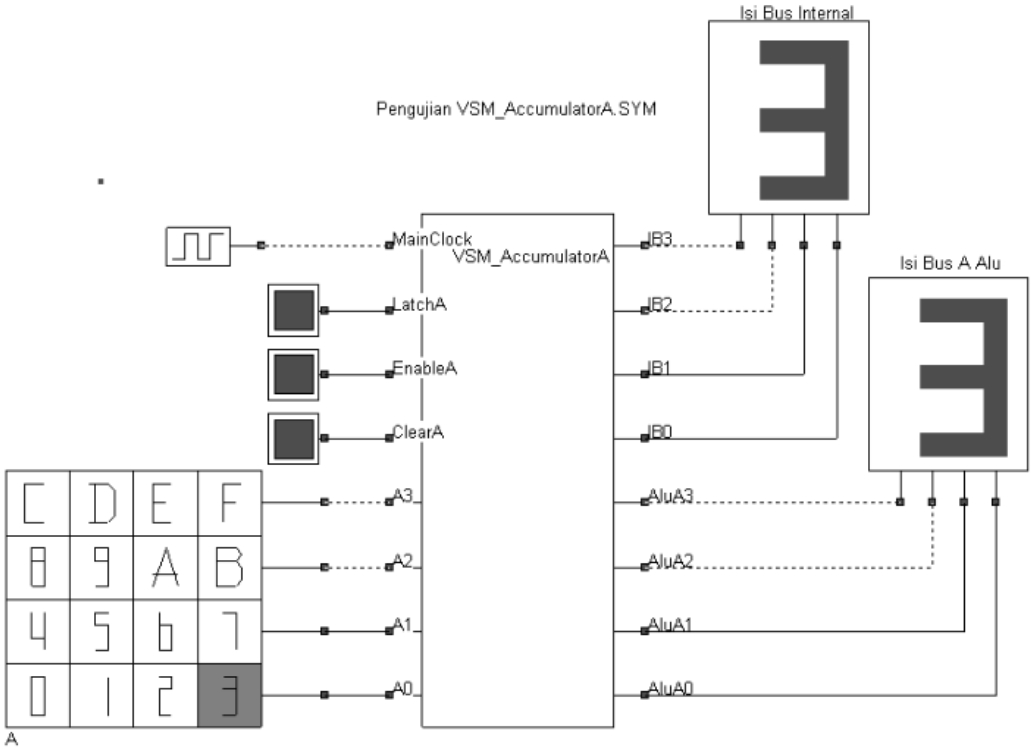
E. PROSEDUR PERCOBAAN

Accumulator A

1. Jalankan DSCH3 dan load modul VSM_AccumulatorA.SCH sehingga diperoleh rangkaian seperti pada gambar 34.
2. Lakukan pengujian *accumulator A* untuk menyimpan data misalnya data 3, dengan memberikan sinyal ClearA=1, LatchA=1, dan klik angka 3 pada keypad. Amati output akumulator! Apakah pada outputnya telah terdapat data 3 atau 0011 biner? Jika ya, maka akumulator telah dapat digunakan untuk menyimpan data.
3. Lakukan pengujian *accumulator A* untuk menyediakan data pada bus internal dengan memberikan sinyal EnableA=1. Amati output akumulator yang terhubung ke bus internal! Apakah pada bus internal telah terdapat data 3 atau 0011 biner? Jika ya, maka akumulator telah dapat menyediakan data pada bus internal.
4. Jika pada tahap pengujian ini rangkaian akumulator telah bekerja dengan baik, simpanlah modul tersebut dalam bentuk simbol dengan nama VSM_AccumulatorA.SYM. Untuk menyimpan rangkaian ini dalam bentuk simbol,

hapuslah terlebih dahulu keypad dan gantilah input akumulator dengan saklar A_3, A_1, A_2, dan A_0.

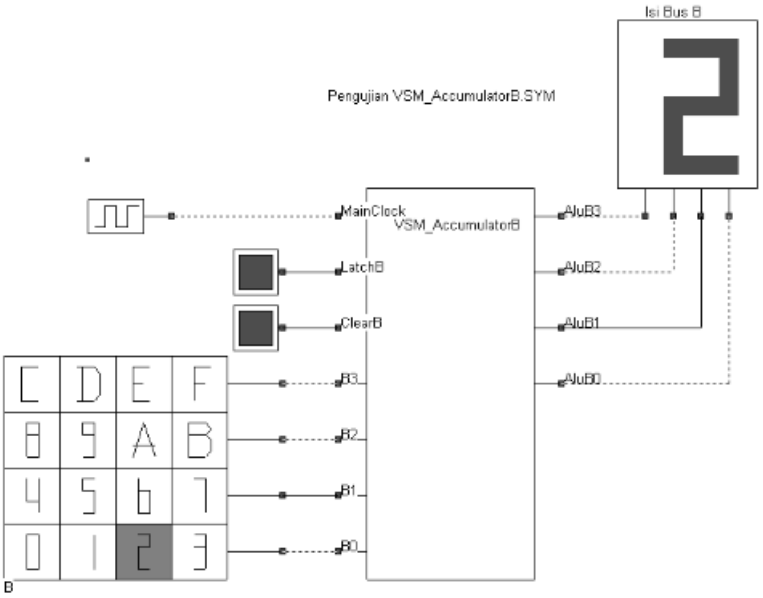
5. Lakukan pengujian terhadap VSM_AccumulatorA.SYM dengan menggunakan rangkaian sebagai berikut:



Gambar 36. Rangkaian pengujian modul VSM_AccumulatorA.SYM

Accumulator B

1. Load modul VSM_AccumulatorB.SCH sehingga diperoleh rangkaian seperti pada gambar 35.
2. Lakukan pengujian *accumulator B* untuk menyimpan data misalnya data 2, dengan memberikan sinyal ClearB=1, LatchB=1, dan klik angka 2 pada keypad. Amati output akumulator! Apakah pada outputnya telah terdapat data 2 atau 0010 biner? Jika ya, maka akumulator telah dapat digunakan untuk menyimpan data.
3. Jika pada tahap pengujian ini rangkaian akumulator telah bekerja dengan baik, simpanlah modul tersebut dalam bentuk simbol dengan nama VSM_AccumulatorB.SYM. Untuk menyimpan rangkaian ini dalam bentuk simbol, hapuslah terlebih dahulu keypad dan gantilah input akumulator dengan saklar B_3, B_1, B_2, dan B_0.
4. Lakukan pengujian terhadap VSM_AccumulatorB.SYM dengan menggunakan rangkaian sebagai berikut.



Gambar 37. Pengujian modul VSM_AccumulatorB.SYM

F. PROSEDUR PERCOBAAN

1. Jelaskan tahap-tahap pembangunan rangkaian akumulator yang dapat menyimpan data!
2. Jelaskan fungsi accumulator A dan accumulator B dalam suatu rangkaian mikroprosesor!

PERCOBAAN VI
VSM (VERY SIMPLE MICROPROCESSOR)
BAGIAN I: UNIT ARITMETIKA DAN INTERKONEKSI DENGAN AKUMULATOR

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat membangun unit artimetika dengan kemampuan operasi ADD dan SUB serta akumulator pendukungnya pada mesin VSM.

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

1. Apa peran utama Unit Aritmetika dalam Very Simple Microprocessor (VSM), dan bagaimaimana unit ini berinteraksi dengan Akumulator?

C. TEORI

VSM merupakan bentuk paling sederhana dari arsitektur mikroprosesor. Percobaan ini akan menitik beratkan pembahasan tentang unit aritmetik dan interkoneksi antara unit aritmetika dan akumulator. Beberapa penjelasan terkait unit-unit pada VSM dijelaskan kembali pada percobaan ini.

Program counter merupakan pencacah 4-bit yang melakukan pencacahan dari 0000 sampai dengan 1111. Outputnya merupakan kode alamat instruksi yang akan diambil dari lokasi memori. Keadaan awal dari *program counter* adalah 0000 yang berarti mikroprosesor mulai bekerja dengan melaksanakan instruksi yang tersimpan pada alamat 0000.

Program memory merupakan memori yang menyimpan program. Masing-masing baris program mempunyai format 8-bit yakni 4-bit pada bagian atas sebagai kode instruksi (*opcode*) dan 4-bit bagian bawah sebagai data yang dioperasikan (operand).

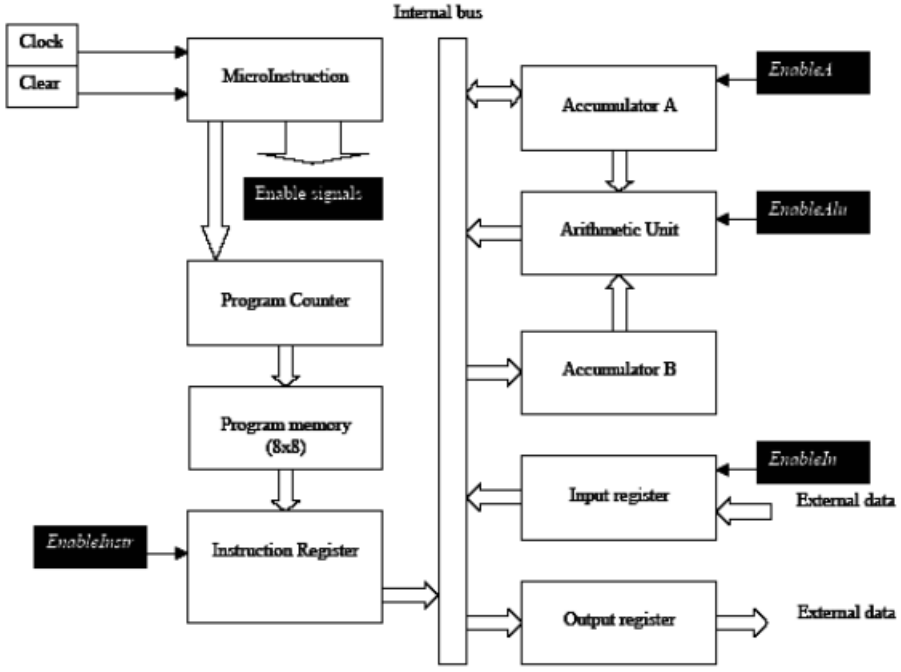
Accumulator A merupakan register 4-bit yang berfungsi menyimpan secara sementara hasil yang diperoleh dari komputasi mikroprosesor. Register ini dilengkapi dengan kontrol *enable*A. Jika *enableA* diberi sinyal tinggi, data yang ada di dalamnya akan ditempatkan pada bus internal.

Accumulator B merupakan register 4-bit yang berfungsi menyimpan data yang akan ditambahkan atau dikurangkan pada operasi penambahan atau pengurangan.

Arithmetic unit merupakan rangkaian aritmetika dengan kemampuan melaksanakan operasi penjumlahan (S=A+B) dan pengurangan (S=A+$\bar{B}$+1).

Input register merupakan register input yang berfungsi menyimpan data dari luar sistem yang akan dimasukkan ke dalam mikroprosessor. Sedangkan *output register* merupakan register yang berfungsi menyimpan data yang akan ditransfer dari bus internal ke luar sistem.

Cara kerja mesin VSM berbasis pada bus internal. Setiap blok memanfaatkan bus dengan menggunakan sinyal *enable*. Sebagai contoh *accumulator A* memiliki sinyal *enableA*, dan jika jika sinyal tersebut bernilai tinggi maka data 4-bit yang dalam *accumulator A* tersebut ditempatkan pada bus internal. Daftar sinyal *enable* ditunjukkan pada gambar 33, dan kontrol dari sinyal-sinyal *enable* tersebutdisediakan oleh blok *microinstruction* yang akan memerankan fungsi yang sangat penting bagi pengendalian mikroprosesor.



Gambar 38. Kontroler membangkitkan sinyal enable yang mengizinkan 1 blok dapat memanfaatkan/mengendalikan bus internal

Tabel 11 mendeskripsikan sinyal *enable* yang diperlukan untuk mengendalikan bus pada setiap blok.

Tabel 11. Deskripsi sinyal enable pada VSM

Sinyal Enable	Deskripsi
EnableA	Menempatkan isi accumulator A ke bus internal
EnableAlu	Menempatkan hasil operasi aritmetika (ADD atau SUB) pada bus internal
EnableInstr	Menempatkan bagian data dari instruksi (4-bit bagian bawah) ke bus internal
EnableIn	Mentransfer data input ke bus internal

1. Instruksi

Instruksi-instruksi yang digunakan pada mesin VSM terdiri atas 4-bit, sehingga dapat memberikan 16 buah instruksi.

No Operation (NOP=0000)

Operasi NOP tidak memiliki pengaruh terhadap register internal. Walaupun demikian, instruksi ini sangat penting untuk mengetahui cara kerja clock dasar dalam mengontrol kerja mikroprosesor.

Addition (ADD=0001)

Instruksi ini memberikan arti isi akumulator ditambahkan dengan data yang ditetapkan sebagai parameter/operand, dan hasilnya disimpan kembali ke akumulator. Penjumlahan dilakukan terhadap data 4-bit dengan mengabaikan *carry*. Contoh: jika A=2, maka instruksi ADD 3 akan memberikan aksi A=A+3 atau A=2+3, dan hasil akhirnya adalah A=5.

Substraction (SUB =0010)

Arti instruksi ini adalah isi akumulator dikurangkan dengan data yang ditetapkan sebagai parameter/operand, dan hasilnya disimpan kembali ke akumulator. Pengurangan dilakukan terhadap data 4-bit dengan mengabaikan *carry*.

Get Input (In=0011)

Arti instruksi ini adalah isi port input ditransfer ke *accumulator* A.

Give Output (OUT=0100)

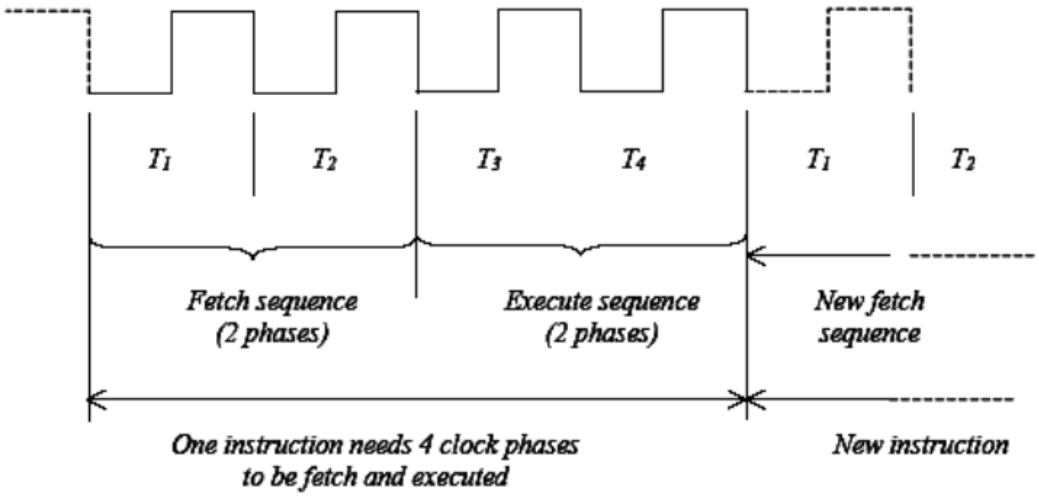
Instruksi ini akan menyebabkan isi *accumulator* A disimpan ke port output. Port output adalah register 4-bit yang menyimpan nilai output dan mempertahankannya sampai isinya diperbaharui dengan instruksi OUT yang baru.

Load Instruction (LDA=0101)

Instruksi ini menyebabkan suatu data yang ditetapkan sebagai parameter/operand diisikan ke accumulator A. Contoh instruksi LDA 9 akan menyebabkan data 9 atau 1001 biner diisikan ke *accumulator* A.

2. Pelaksanaan Instruksi

Mekanisme pelaksanaan perintah mesin VSM ini berbasis pada mikroinstruksi internal. Mikroinstruksi merupakan kumpulan sinyal *enable* yang dibangkitkan pengontrol. Pelaksanaan dari masing-masing instruksi berbasis pada empat fase waktu yang berbeda seperti diilustrasikan pada gambar 35.



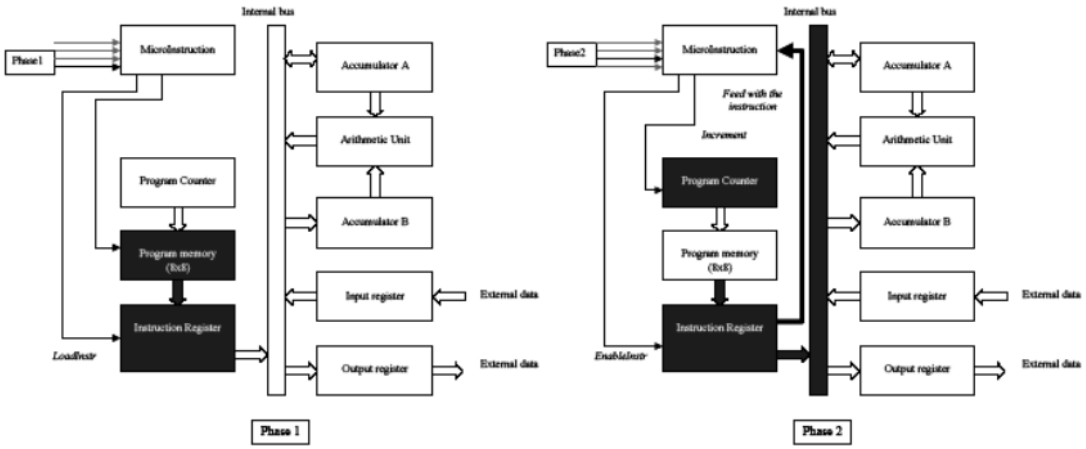
Gambar 39. Fase pelaksanaan instruksi

Penjelasan masing-masing fase dari pelaksanaan instruksi ditunjukkan pada tabel 12 berikut ini.

Tabel 12. Fase pelaksanaan instruksi

Fase	Nama	Deskripsi
Fase 1	Address State	Isi memori dimuatkan ke register instruksi
Fase 2	Increment State	Alamat pada *program counter* ditambah 1 (*increment*). Output register instruksi mengumpan input *decoder* mikroinstruksi sehingga membangkitkan sinyal enable.
Fase 3	Execute Step 1	Mikroprosesor melaksanakan instruksi step pertama
Fase 4	Execute Step 2	Mikroprosesor melaksanakan instruksi step kedua

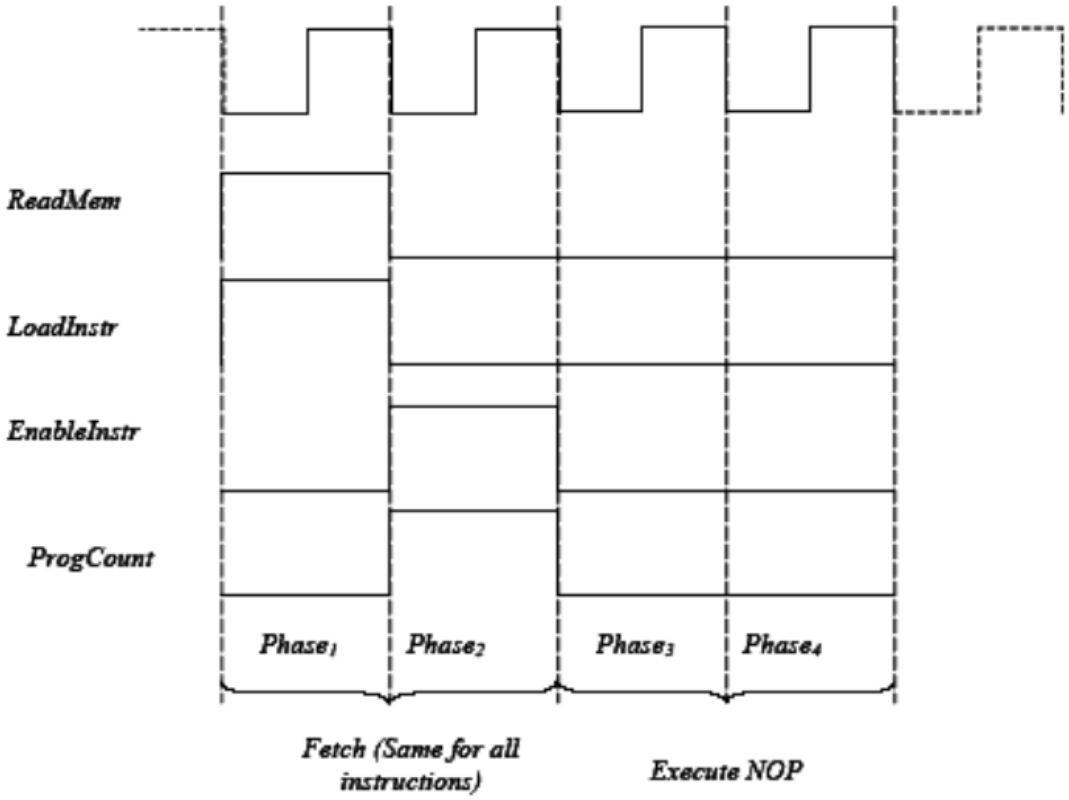
Pada gambar 40 ditunjukkan ilustrasi fase pengambilan instruksi dari memori dan selanjutnya ditempatkan pada bus internal. Pada fase 1, input alamat memori diberi umpan dari output program counter dan pengontrol mikroinstruksi membangkitkan sinyal *enable* ReadMem, dan LatchInstrReg (LoadInstr) sehingga isimemori dimuatkan ke register instruksi. Pada fase 2, output *program counter* bertambah 1 (*increment*) dan pengontrol membangkitkan sinyal *EnableInstr* sehingga bagian data dari instruksi dimuat ke bus internal. Pada fase ini output register instruksi diumpankan ke input pengontrol mikroinstruksi untuk membangkitkan sinyal *enable* yang diperlukan pada fase berikutnya.



Gambar 40. Ilustrasi fase pengambilan instruksi (fetch cycle) yakni fase 1 dan fase 2

No Operation (NOP=0000)

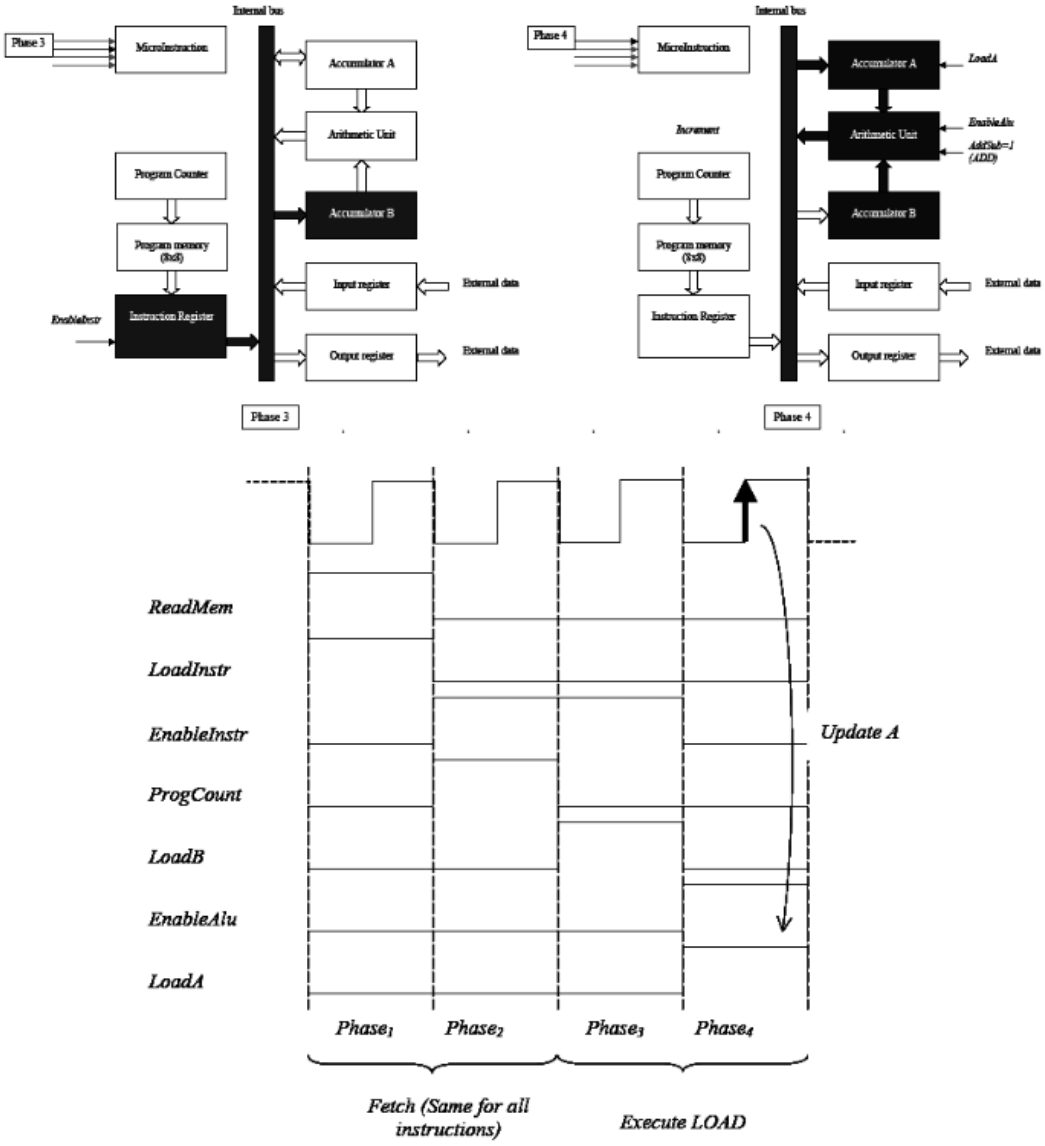
Selama fase 1 pengambilan instruksi adalah urut-urutan akses memori (ReadMem=1), pemuatan instruksi ke register instruksi (LoadInstr). Sedangkan pada fase 2, instruksi pada register instruksi dikirim ke pengontrol mikroinstruksi dan sambil program counter ditambah 1. Setelah fase pengambilan diteruskan dengan fase pelaksanaan instruksi. Jika pada fase ini dijumpai instruksi NOP maka tidak akan menghasilkan aktivitas apapun



Gambar 41. Urut-urutan ambil dan laksanakan pada instruksi NOP

Addition (ADD=0001)

ADD adalah operasi penambahan antara isi *accumulator A* dengan data operand 4-bit. Data operand ditempatkan terlebih dahulu pada *accumulator B* (fase 3) kemudian Unit Aritmetika menjumlahkan isi *accumulator A* dan *accumulator B* dan hasilnya disimpan pada *accumulator A* (fase 4). Ilustrasi fase 3 dan fase 4 pelaksanaan instruksi ADD ditunjukkan pada gambar 42.



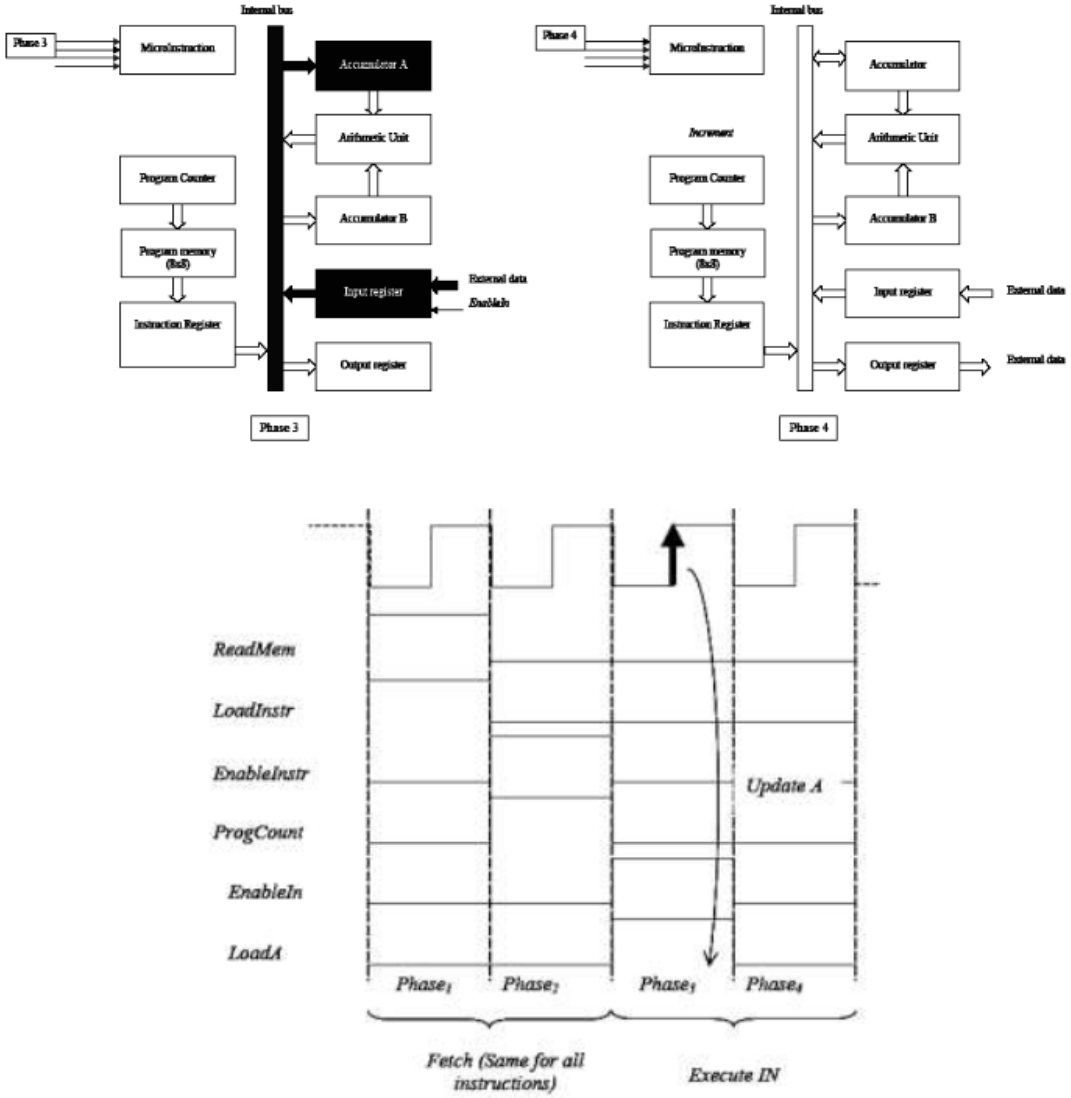
Gambar 42. Ilustrasi pelaksanaan instruksi ADD

Substraction (SUB=0010)

Fase pelaksanaan instruksi SUB sama dengan instruksi ADD, perbedaannya sinyal AddSub=1 agar Unit Aritmetika melaksanakan instruksi SUB.

Get Input (In=0011)

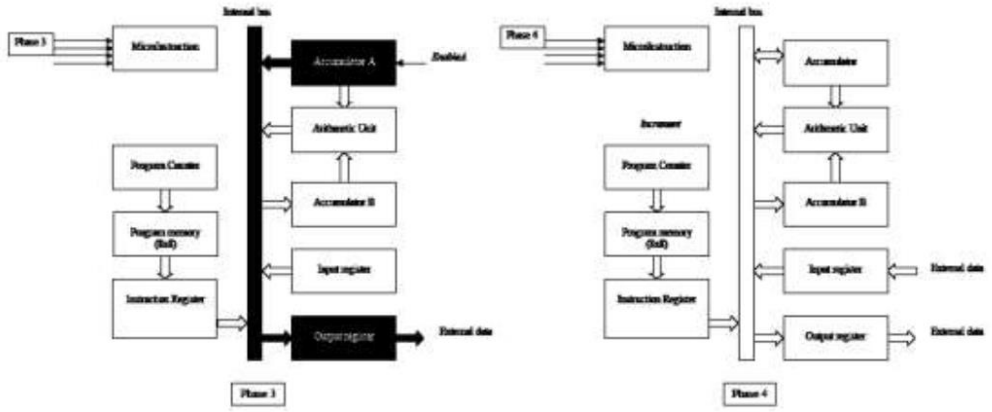
Isi port input ditransfer ke accumulator A selama fase 3, sedangkan pada fse ke 4 tidak terjadi aktivitas apapun. Ilustrasi pelaksanaan instruksi In ditunjukkan pada gambar 43.



Gambar 43. Ilustrasi pelaksanaan instruksi In

Give Out (Out=0100)

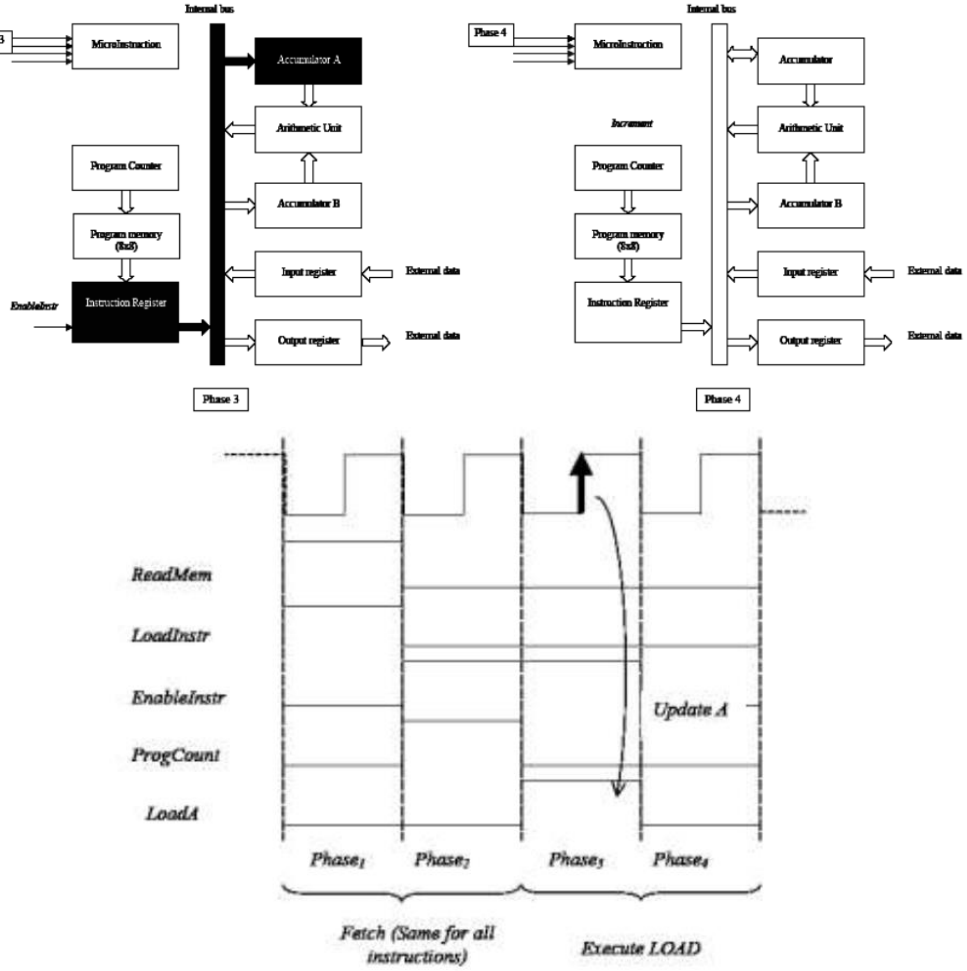
Isi *accumulator A* ditransfer ke port output lewat bus internal selama fase 3. Port output menyimpan nilai *accumulator* A dan mempertahankannya sampai isinya diperbaharui dengan instruksi OUT yang baru. Pada fase ke 4, mikroprosesor tidak aktif seperti diilustrasikan pada gambar 44.



Gambar 44. Ilustrasi pelaksanaan instruksi Out

Load Instruction (LDA=0101)

Instruksi ini akan mentransfer data 4-bit pada bagian operand instruksi LDA ke accumulator A. Contoh instruksi LDA 9 akan menyebabkan data 9 atau 1001 biner diisikan ke *accumulator A*. Ilustrasi pelaksanaan perintah LDA ditunjukkan pada gambar 45.



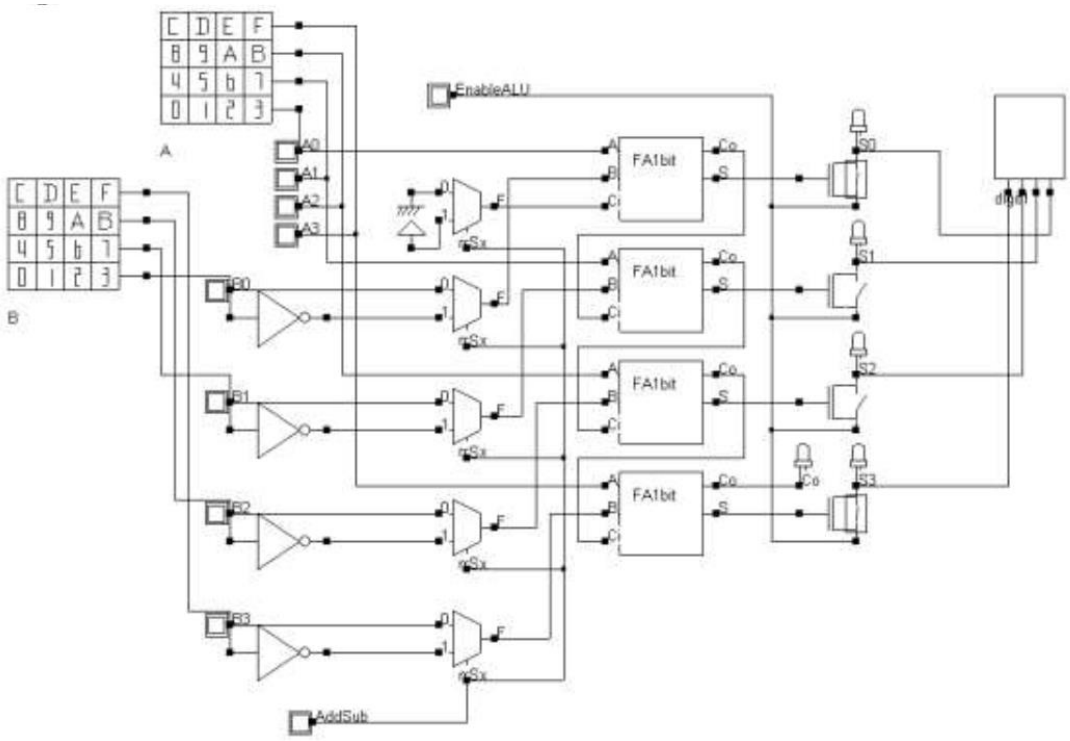
Gambar 45. Ilustrasi pelaksanaan perintah LDA

3. Rancangan Blok Dasar Mesin VSM (bagian 2)

Unit Aritmetika

Rancangan unit ini berbasis pada rangkaian *full adder* 1-bit, dan agar dapat menangani data dengan panjang 4-bit, maka perlu disediakan 4 buah *full adder* 1-bit yang disusun secara berjajar membentuk konfigurasi *full adder* paralel 4-bit..

Oleh karena unit aritmetika pada VSM ini dirancang dapat melakukan dua operasi yakni penjumlahan (ADD) dan pengurangan (SUB), maka *full adder* paralel 4-bit perlu dilengkapi rangkaian tambahan agar dapat melakukan kedua operasi itu. Agar *full adder* dapat melakukan operasi SUB (B-A), dibutuhkan rangkaian yang dapat membangkitkan komplemen ke-2 dari A dan mengatur agar *previous carry* sama dengan 1. Implementasi rangkaian untuk kedua operasi tersebut ditunjukkan pada gambar 46.

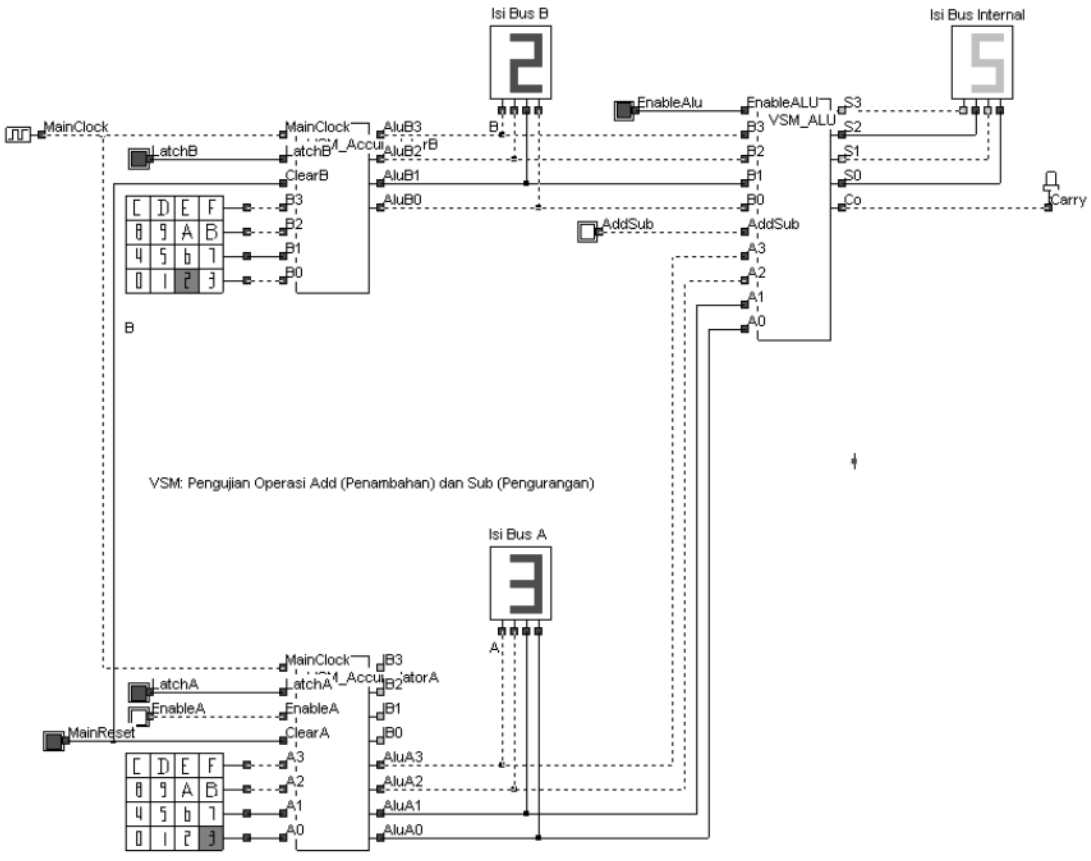


Gambar 46. Struktur unit aritmetika dengan kemampuan operasi ADD dan SUB

Pada gambar 46 ditunjukkan bahwa input B *full adder* berasal dari dua sumber yakni data B dan komplemennya ($\bar{B}$). Demikian pula input Ci berasal dari dua sumber yakni data 0 dan data 1. Untuk memilih salah satu dari dua sumber tersebut, pada input B dan input Ci dipasang multiplekser 2 ke1. Pemilih multiplekserditetapkan sebagai pemilih operasi yakni AddSub. Jika AddSub=0 maka data B dipilih sebagai input B *full adder*, dan data 0 sebagai input Ci, dengan demikian *full adder* melakukan operasi ADD (A+B). Jika AddSub=1 maka data $\bar{B}$ dipilih sebagai input B *full adder*, dan data 1 sebagai input Ci, dengan demikian *full adder* melakukan operasi SUB (A+ $\bar{B}$ +1=A-B).

Dengan menggunakan dua buah akumulator dan rangkaian aritmetika tersebut dapat disusun suatu rangkaian paling mendasar dari suatu mikroprosesor. Interkoneksi akumulator dengan unit aritmetika ditunjukkan pada gambar 47. Rangkaian tersebut telah dapat digunakan untuk mendemonstrasikan proses penjumlahan dua buah bilangan sebagai berikut:

a. Matikan fungsi reset pada kedua akumulator dengan memberikan sinyal MainReset=1.
b. Isilah *accumulator A* dengan data yang dikehendaki misalnya 3, dengan memberikan sinyal *LatchA=1*, dan klik angka 3 pada keypad.
c. Isilah *accumulator B* dengan data yang dikehendaki misalnya 2, dengan memberikan sinyal *LatchB=1*, dan klik angka 2 pada keypad.
d. Berikan AddSub=0 pada Unit Aritmetika agar unit tersebut melakukan operasi penjumlahan. Untuk menampilkan hasil operasi pada bus internal, berikan EnableAlu=1. Dengan cara tersebut, pada bus internal akan ditampilkan angka 5 yang merupakan hasil penjumlahan 2+3. Jika diinginkan unit aritmetika melakukan operasi pengurangan, berikan AddSub=1.



Gambar 47. Koneksi antara accumulator A, accumulator B dan unit aritmetika untuk pengujian instruksi ADD dan SUB

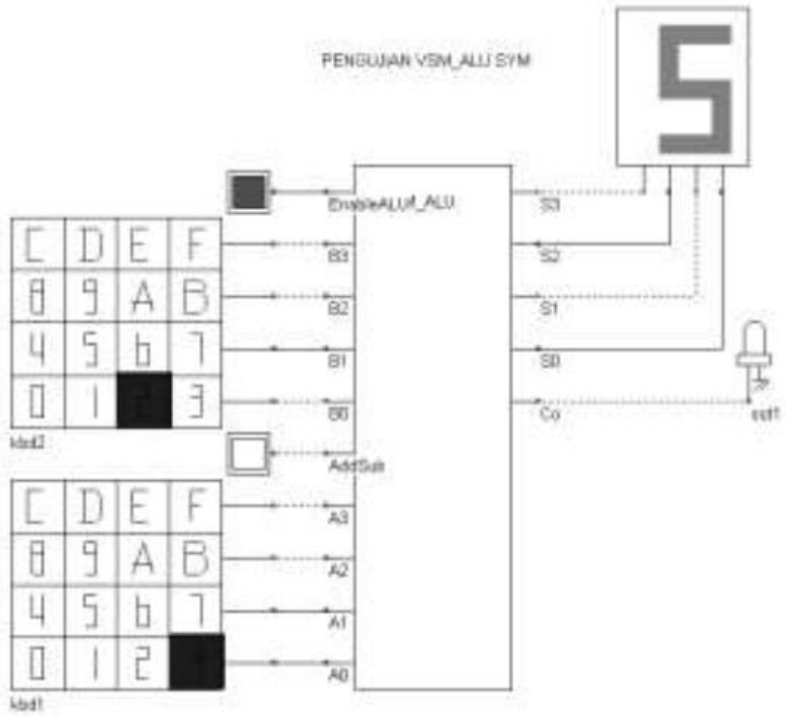
D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul VSM_AccumulatorA.SCH, Modul VSM_AccumulatorB.SCH, Modul VSM_ALU.SCH, dan Modul VSM_RegARegBAlu.SCH.

E. PROSEDUR PERCOBAAN

Unit Aritmetika

1. Load modul VSM_Alu.SCH sehingga diperoleh rangkaian seperti pada gambar 46.
2. Lakukan pengujian unit ini untuk menjumlah bilangan 3 dan 2 dengan klik pada angka 3 pada keypad A dan angka 2 pada keypad B. Selanjuntya berikan AddSub=0 dan EnableAlu=1. Apakah output unit ini telah menghasilkan angka 5 yang merupakan hasil penjumlahan 3 dan 2? Jika ya, maka unit ini telah dapat bekerja dengan baik sebagai penjumlah.
3. Berikan AdSub=1! Apakah unit ini dapat melakukan pengurangan?
4. Jika hasil pengujian telah dapat menunjukkan bahwa unit ini dapat melakukan penjumlahan dan pengurangan, simpanlah unit ini dalam bentuk simbol dengan nama VSM_Alu.SYM. Ingat! Sebelum anda menyimpannya ke dalam bentuk simbol, hapus terlebih dahulu keypad pada input A dan B serta gantilah input A dengan saklar $A_3A_2A_1A_0$ dan input B dengan saklar $B_3B_2B_1B_0$. Hapus pulaperaga 7 segmen pada output unit ini dan gantilah dengan LED $S_3S_2S_1S_0$.
5. Lakukan pengujian VSM_ALU.SYM dengan menggunakan rangkaian berikut ini:



Gambar 48. Pengujian modul VSM_ALU.SYM

Interkoneksi Unit Aritmetika dan Akumulator

1. Susun rangkaian interkoneksi unit aritmetika dengan akumulator-akumulator pendukungnya seperti gambar 47. Rangkaian ini juga sudah tersedia dalam file VSM_RegARegBAlu.SCH. Jika anda tidak menginginkan merangkai sendiri, load file VSM_RegARegBAlu.SCH.
2. Lakukan pengujian interkoneksi ini seperti penjelasan gambar 45 di atas.

F. TUGAS AKHIR

1. Jelaskan tahap-tahap pembangunan rangkaian unit aritmetika yang dapat melakukan operasi penjumlahan dan pengurangan!
2. Tuliskan sinyal kontrol yang diperlukan untuk melakukan penjumlahan bilangan 2 dan 3 pada rangkaian unit aritmetika yang dilengkapi dengan *accumulator* A dan *accumulator* B!

PERCOBAAN VII
VSM (VERY SIMPLE MICROPROCESSOR)
BAGIAN III: UNIT INPUT

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat membangun unit input dan output pada mesin VSM dan menjelaskan cara kerja mikroprosesor dalam melakukan operasi transfer data dari unit input ke akumulator.

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

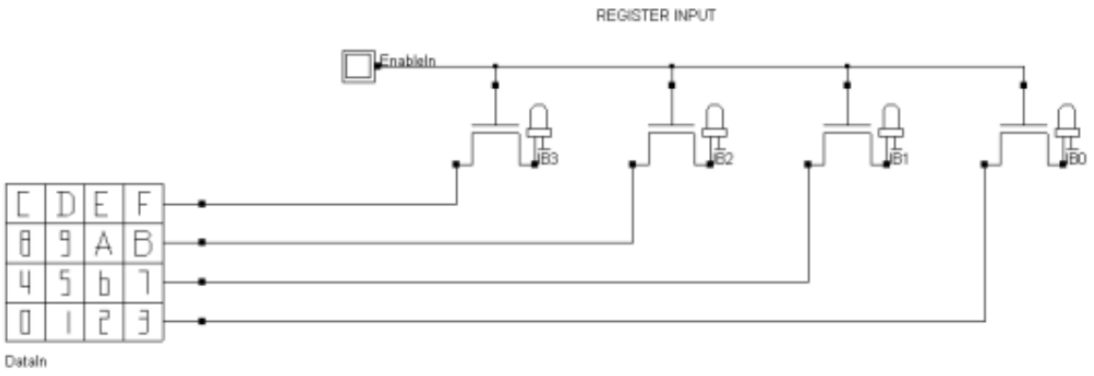
1. Apa peran utama Unit Input dalam Very Simple Microprocessor (VSM), dan bagaimana unit ini digunakan dalam proses komunikasi dengan perangkat luar?

C. TEORI

Selain unit aritmetika dan akumulator, unit lain yang penting dalam VSM adalah unit input dan unit output. Unit input digunakan untuk mentransfer data dari periferal input seperti *keypad* ke bus internal dan selanjutnya disimpan ke akumulator. Dalam VSM, unit input berupa register yang dilengkapi dengan pengontrol EnableIn. Jika EnableIn=1 maka data dari periferal input akan ditransfer ke bus internal, dan sebaliknya jika EnableIn=0, unit input dalam keadaan *inhibit* yakni tidak mentransfer data.

Register Input

Register input dibangun dari 4 buah *tri-state buffer* seperti ditunjukkan pada gambar 49. Dalam hal ini sel register input tidak menggunakan flip-flop D karena data input langsung ditransfer *ke accumulator A*



Gambar 49. Register input

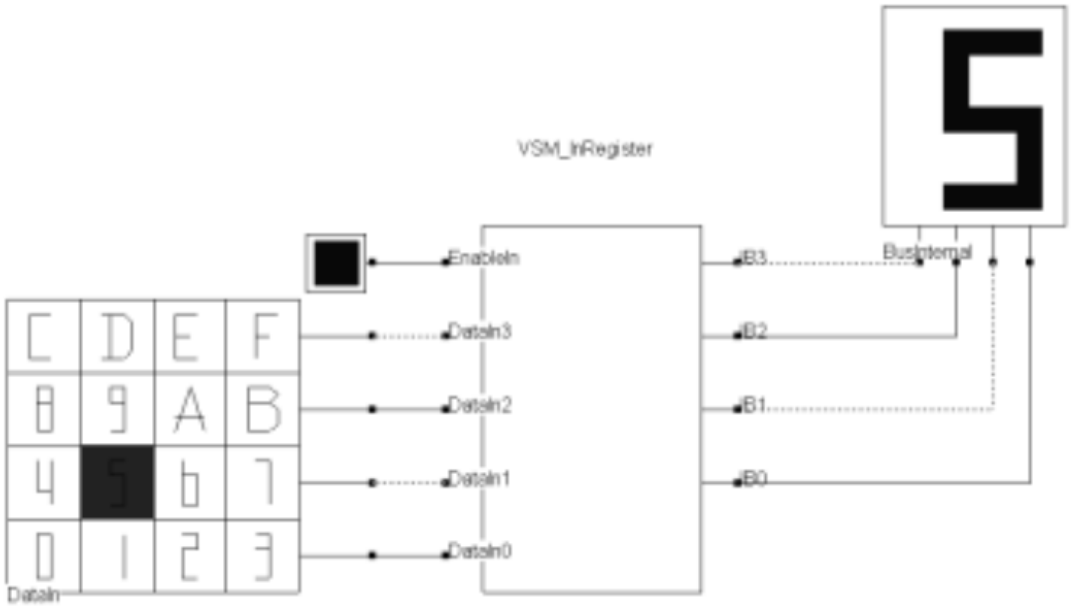
D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul VSM_RegARegBAlu_Versi2.SCH, modul VSM_InRegister.SCH.

E. PROSEDUR PERCOBAAN

Unit Input

1. Load file VSM_InRegister.SCH sehingga diperoleh rangkaian seperti pada gambar 49.
2. Lakukan pengujian unit input dalam mentransfer data dari periferal input berbentuk keypada ke bus internal.
3. Jika hasil pengujian menunjukkan bahwa unit input telah dapat bekerja dengan baik, simpanlah ke dalam bentuk simbol dengan nama VSM_InRegister.SYM. Sebelum anda menyimpan ke dalam bentuk simbol, hapuslah terlebih dahulu keypad pada rangkaian tersebut dan gantilah dengan input saklar DataIn3, DataIn2, DataIn1, dan DataIn0.
4. Lakukan pengujian terhadap simbol VSM_InRegister.SYM dalam mentrasfer data dari keypad ke bus internal menggunakan rangkaian berikut ini.



Gambar 50. Rangkaian untuk menguji modul VSM_InRegister.SYM

F. TUGAS AKHIR

1. Jika sebuah pulsa clock dianggap sama dengan 1 fase pelaksanaan mikrooperasi, gambarkan fase-fase pemberian sinyal-sinyal kontrol pada operasi di atas dengan menggunakan visualisasi sinyal clock yang diberikan!

PERCOBAAN VIII
VSM (VERY SIMPLE MICROPROCESSOR)
BAGIAN IV: UNIT OUTPUT DAN INTERKONEKSI DENGAN UNIT ARITMETIKA

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat membangun unit input dan output pada mesin VSM dan menjelaskan cara kerja mikroprosesor dalam melakukan operasi transfer data dari unit input ke akumulator dan dari akumulator ke unit output.

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

1. Apa peran utama Unit Output dalam Very Simple Microprocessor (VSM), dan bagaimana unit ini berinteraksi dengan Unit Aritmetika serta Akumulator?

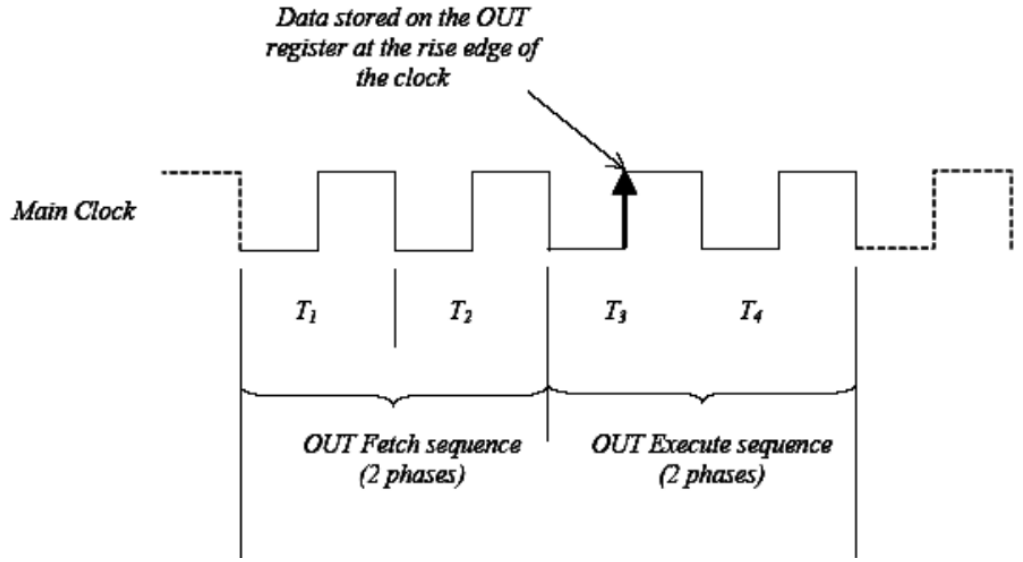
C. TEORI

Selain unit aritmetika dan akumulator, unit lain yang penting dalam VSM adalah unit input dan unit output. Unit output digunakan untuk mentransfer data dari bus internal ke periferal output seperti peraga 7-segmen.

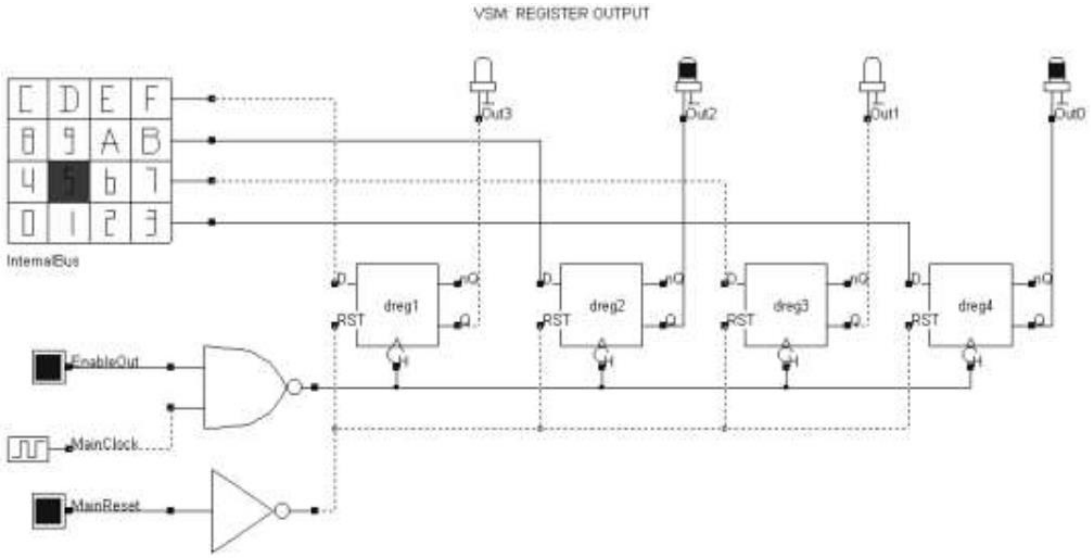
Dalam VSM, unit output juga merupakan suatu register yang dilengkapi dengan pengontrol LoadOut dan MainReset. Unit output akan mentransfer data dari bus internal ke periferal output jika LoadOut=1 dan MainReset=1.

Register Output

Sel-sel register output dibangun dari flip-flop D seperti ditunjukkan pada gambar 51. Register ini dipersyaratkan dapat bekerja yakni menyimpan data yang ada pada inputnya pada saat munculnya tepi positif dari pulsa clock. Syarat ini harus diperhatikan karena jika ada operasi OUT pada fase ke-3, operasi tersebut dilakukan pada tepi positif dari clock dan bukan pada tepi negatif seperti ditunjukkan pada gambar 49. Untuk memenuhi persyaratan ini, maka input clock dari sel-sel flip-flop D dihubungkan dengan gerbang NAND sehingga rangkaian menjadi sensitif terhadap tepi naik atau tepi positif dari clock.



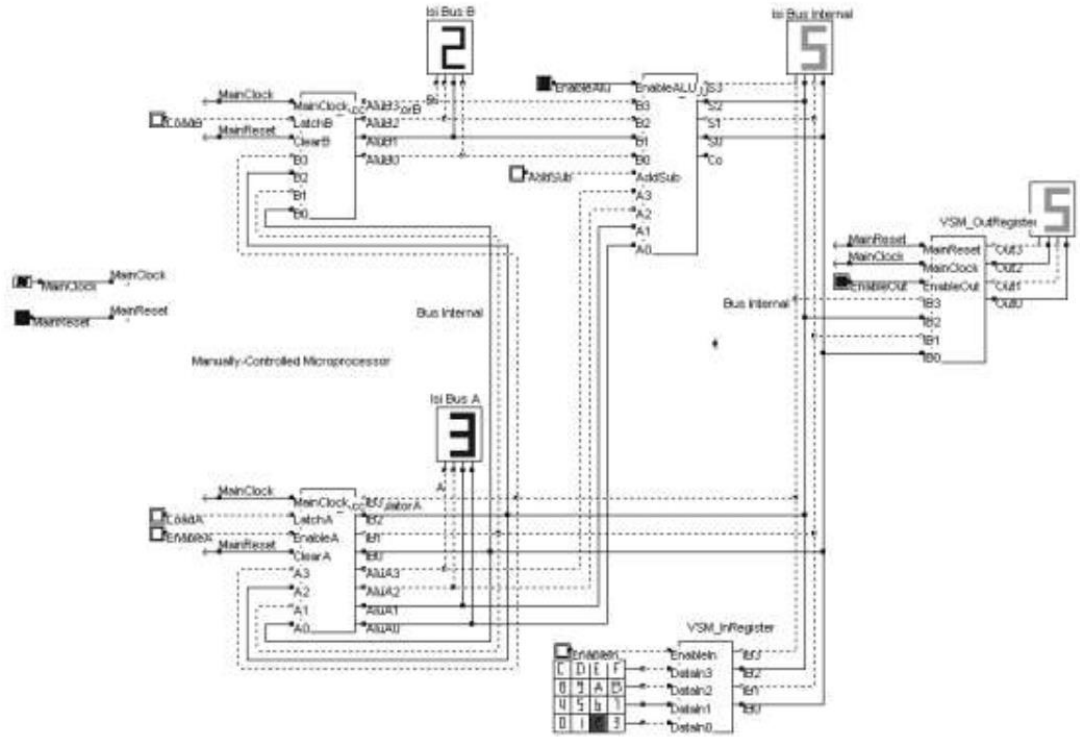
Gambar 50. Register output harus menyimpan data pada tepi naik clock dari fase ke-3



Gambar 51. Register output

Interkoneksi Unit Input/Output Dengan Unit Aritmetika

Interkoneksi unit input/output dengan unit aritmetika ditunjukkan pada gambar 52. Pada rancangan ini, digunakan tanda panah (*arrow*) untuk menghubungkan titik-titik yang sama. Tanda panah pada DSCH2/DSCH3 memiliki fungsi untuk menyederhanakan koneksi. Jika suatu titik pada rangkaian dihubungkan dengan tanda panah yang memiliki nama sama dengan tanda panah yang terhubung dengan titik yang lain maka titik-titik itu dianggap terhubung menjadi satu. Misalnya suatu panah diberi nama MainClock dihubungkan dengan sumber clock, maka semua titik yang terhubung dengan tanda panah bertulisan MainClock akan dianggap tersambung dengan sumber clock.



Gambar 52. Mikroprosesor manual

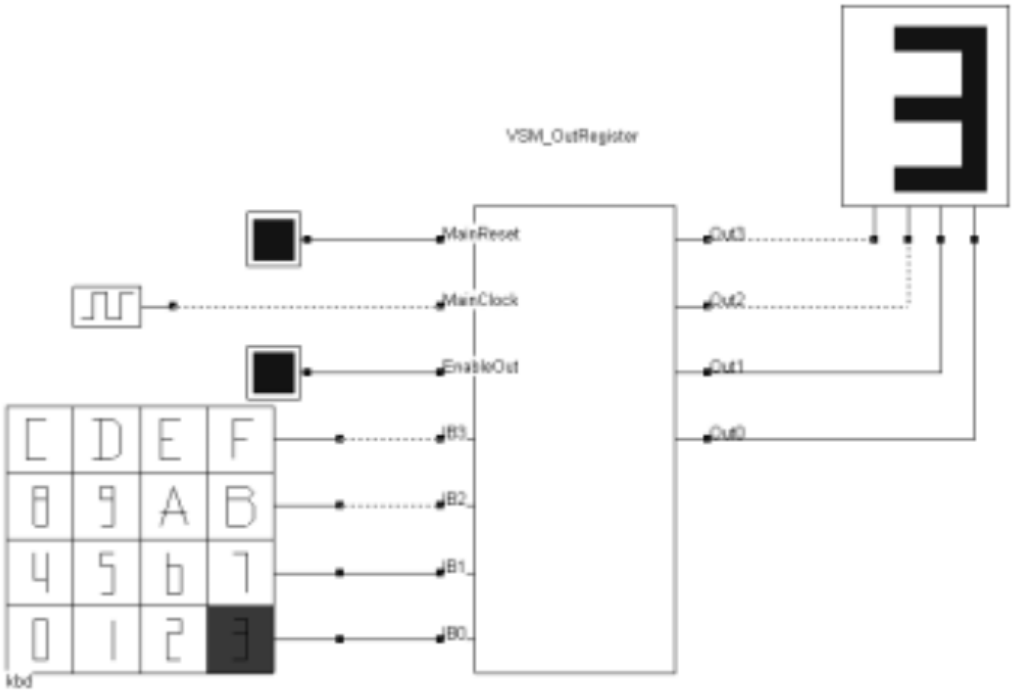
D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul VSM_RegARegBAlu_Versi2.SCH, modul VSM_InRegister.SCH, dan modul VSM_OutRegister.SCH.

E. PROSEDUR PERCOBAAN

Unit Output

1. Load file VSM_OutRegister.SCH sehingga diperoleh rangkaian seperti pada gambar 51.
2. Lakukan pengujian unit output dalam mentransfer data dari bus internal ke periferal output peraga 7-segmen.
3. Jika hasil pengujian menunjukkan bahwa unit output telah dapat bekerja dengan baik, simpanlah ke dalam bentuk simbol dengan nama VSM_OutRegister.SYM. Sebelum anda menyimpan ke dalam bentuk simbol, hapuslah terlebih dahulu keypad pada rangkaian tersebut dan gantilah dengan input saklar IB3, IB2, IB1, dan IB0.
4. Lakukan pengujian terhadap simbol VSM_OutRegister.SYM dalam mentrasfer data dari bus internal ke peraga 7-segmen menggunakan rangkaian berikut ini.



Gambar 54. Rangkaian untuk menguji modul VSM_OutRegister.SYM

Interkoneksi Unit Input/Output Dengan Unit Aritmetika

1. Load modul VSM_RegARegBAlu_Versi2.SCH , dan lengkapi rangkaian tersebut dengan unit input/output sehingga menghasilkan rangkaian seperti pada gambar 52.
2. Lakukan pengujian untuk mentransfer data 3 dari unit input ke unit output.
3. Lakukan pula pengujian untuk mentransfer data 3 dari unit input ke accmulator A, data 2 dari unit input ke accumulator B, menjumlahkan isi accumulator A dan accumulator B, dan hasilnya disimpan ke accumulator A dan dikirim ke unit output.

F. TUGAS AKHIR

1. Sebutkan sinyal kontrol yang diperlukan untuk:
   a. mentransfer data dari unit input ke unit output!
   b. mentransfer data dari unit input ke accumulator A, dari unit input ke accumulator B, menjumlahkan isi accumulator A dan B, menyimpan hasil penjumlahan ke accumulator A, dan mengirim hasil penjumlahan ke unit output?

PERCOBAAN IX
VSM (VERY SIMPLE MICROPROCESSOR)
BAGIAN V: UNIT MEMORI, COUNTER, DAN REGISTER INSTRUKSI

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat membangun unit memori dan kontrol pada mesin VSM.

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

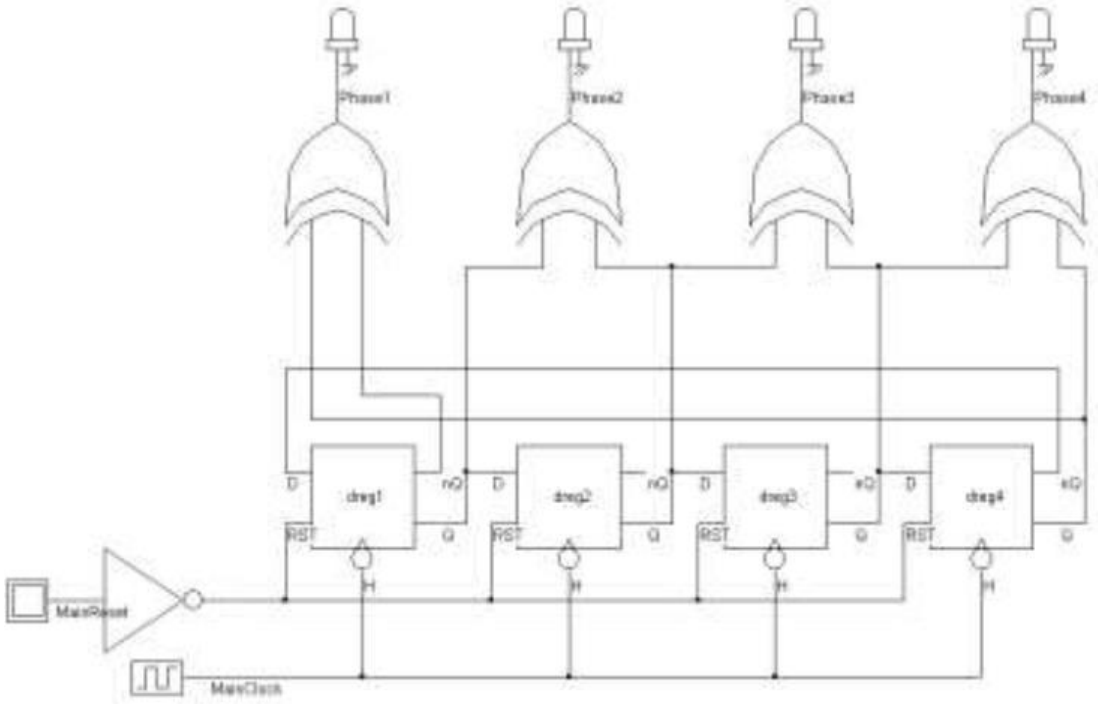
1. Apa peran utama Unit Memori dalam VSM, dan bagaimana unit ini digunakan untuk menyimpan program dan data?
2. Apa fungsi utama Counter dalam VSM, dan bagaimana unit ini digunakan dalam menghitung instruksi atau mengatur aliran program?
3. Apa peran utama Register Instruksi dalam VSM, dan bagaimana unit ini digunakan untuk menyimpan instruksi yang akan dieksekusi??

C. TEORI

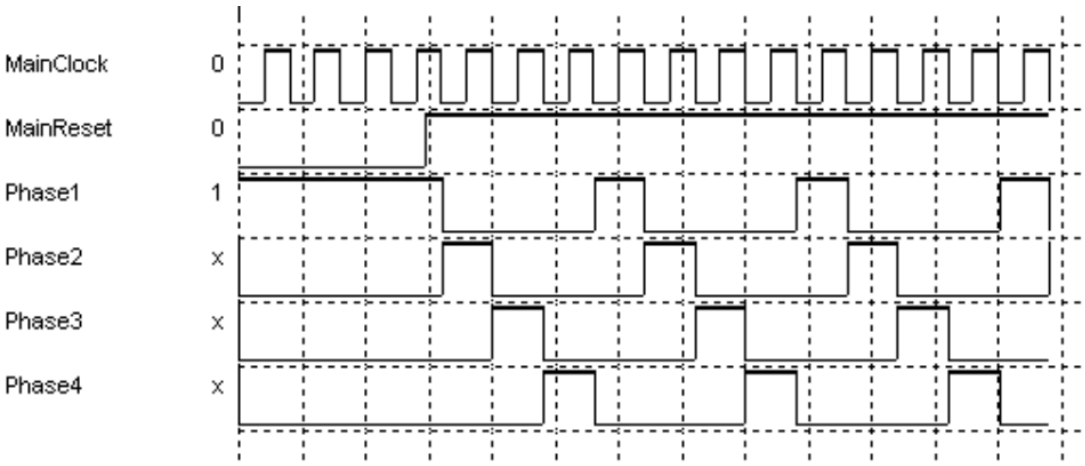
Pada praktikum yang lalu telah dipelajari cara kerja mikroprosesor manual yang instruksi-instruksinya diberikan melalui penekanan tombol saklar. Pada praktikum ini akan dibangun mikroprosesor yang berkerja berbasis pada program yang diberikannya (*fully programmable microprocessor*). Untuk membangun mikroprosesor ini diperlukan rangkaian yang dapat membangkitkan sinyal kontrol yang diperlukan dalam operasi mikroprosesor. Oleh karena mesin VSM ini dalam melaksanakan instruksinya berbasis pada pelaksanaan instruksi 4 fase, maka langkah pertama yang perlu dilakukan adalah membangun terlebih dahulu rangkaian generator fase yang dapat membangkitkan sinyal fase1, fase2, fase3, dan fase4 setiap terjadinya tepi turun atau negatif dari clock.

Generator Fase

Generator fase pada mesin ini merupakan ring counter yang dibangun dengan menggunakan 4 buah flip-flop D dan 4 buah gerbang XOR seperti ditunjukkan pada gambar 55.



Gambar 55. Ring counter sebagai generator fase

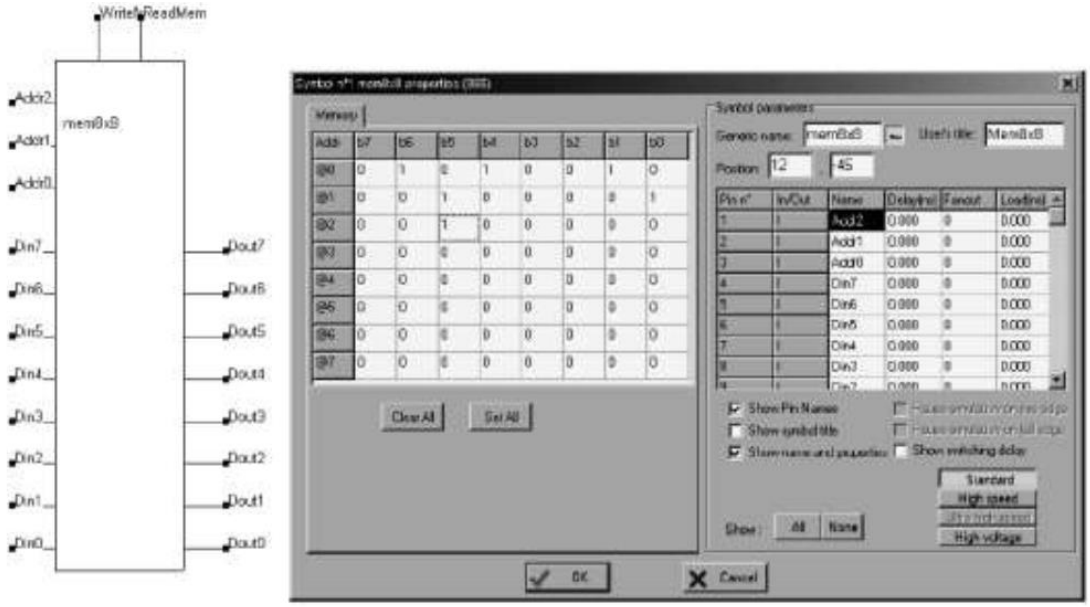


Gambar 56. Watak ring counter hasil simulasi

Generator fase ini dilengkapi dengan Clear jenis *active-low*. Jika MainReset=0, generator tidak membangkitkan urut-urutan sinyal. Agar generator fase dapat bekerja dengan normal input clear diberi sinyal tinggi (MainReset=1).

Memori

Memori yang digunakan pada VSM ini adalah memori ukuran 8X8-bit, yang dapat diperoleh dari *basic symbol palette* dari DSCH3.



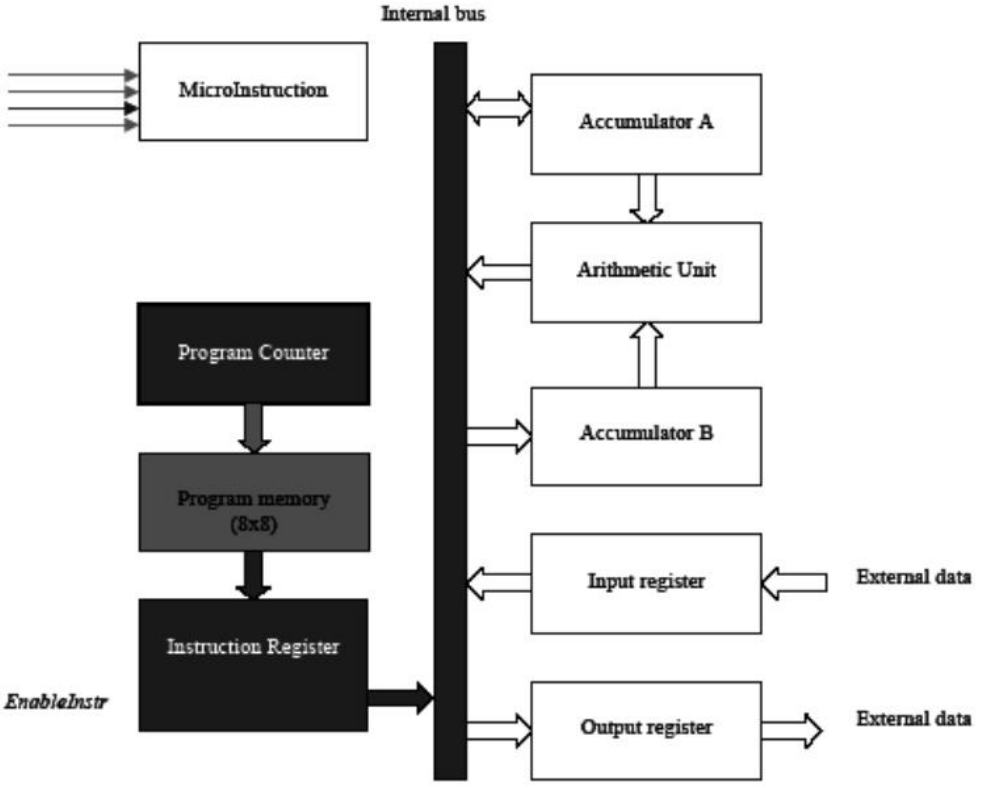
Gambar 57. Memori 8X8-bit yang digunakan untuk menyimpan instruksi

Cara memrogram memori ini sangat mudah. Dengan melakukan klik double pada simbol memori, dapat diperoleh tampilan sel-sel memori yang siap diganti/dimodifikasi. Untuk memodifikasi isi sel memori, tinggal melakukan klik pada data yang akan dimodifikasi. Pada gambar 57 ditunjukkan cara menyimpan data 0101 0010 biner atau 52 heksadesimal pada alamat 0, data 0010 0001 biner atau 21 heksadesimal pada alamat 1, data 0010 0000 biner atau 20 heksadesimal pada alamat 2, dan data 0000 0000 biner atau 00 heksadesimal pada alamat 3.

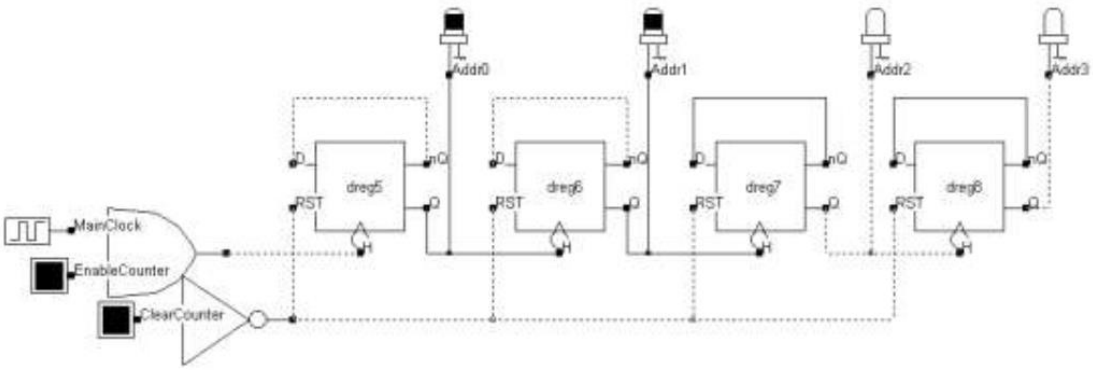
Program Counter

Output program counter merupakan kode-kode alamat dari instruksi yang akan diambil. Misalnya akan diambil instruksi pada alamat 2, maka program counter menyediakan kode 2 yakni 0010 pada outputnya, dan selanjutnya output ini diumpankan ke pemilih alamat pada memori. Setelah suatu instruksi selesai diambil, isi program counter bertambah 1 (increment) yang merupakan alamat instruksi berikutnya yang akan diambil. Pada saat start isi program counter adalah 0000.

Oleh karena memorinya berukuran 8 alamat yang memerlukan 3-bit pemilih alamat maka program counter yang disediakan minimal memiliki output 3-bit (modulo-8). Pada rancangan ini, program counter yang dibangun merupakan counter modulo-16 yang menyediakan output dengan panjang 4-bit. Rangkaiannya terdiri atas 4 buah flip-flop D yang membentuk fungsi *toggle* (berwatak seperti flip-flop T) dan merupakan *asynchronous counter* seperti ditunjukkan pada gambar 59.



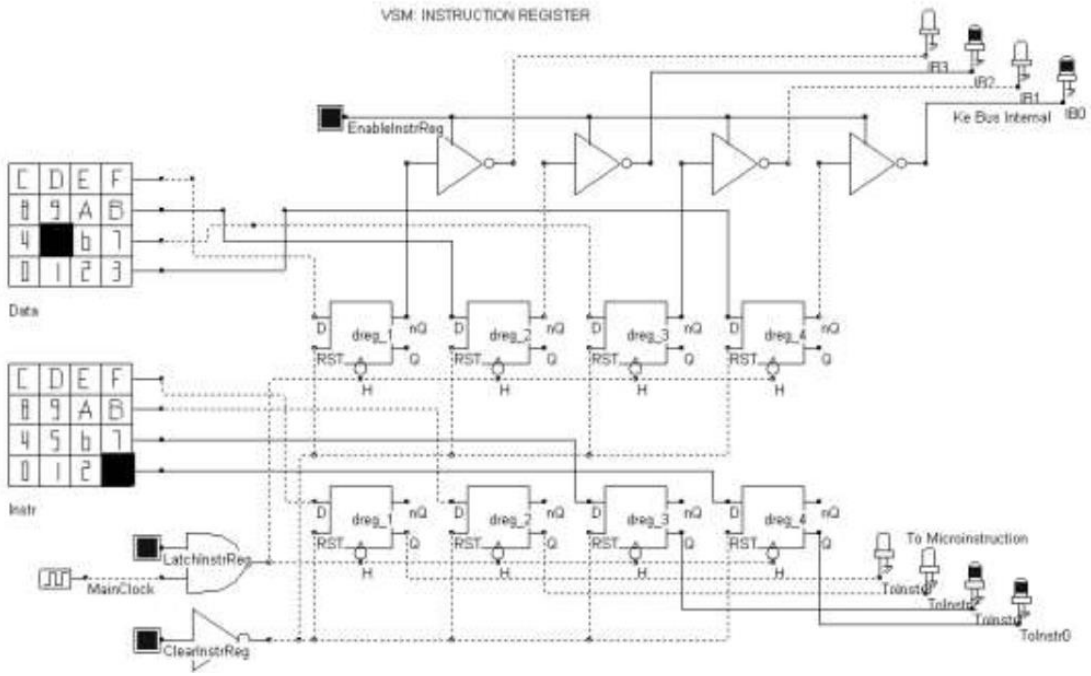
Gambar 58. Output program counter mengumpan pemilih alamat pada memori



Gambar 59. Rangkaian program counter

Register Instruksi

Register instruksi berfungsi menyimpan instruksi yang diambil dari memori. Instruksi yang tersimpan dalam memori terbagi menjadi 2 bagian yakni bagian *opcode* menempati 4-bit orde tinggi dan *operand* atau data menempati 4-bit orde rendah. *Opcode* sepanjang 4-bit dalam register instruksi tersimpan 4 buah flip-flop D yang outputnya langsung dapat mengumpan rangkaian dekoder mikroinstruksi. Sedangkan data 4-bit dalam register instruksi disimpan pada 4 buah flip-flop D yang terpisah dan dapat ditransfer ke bus internal dengan memberi EnableInstrReg=1.



Gambar 60. Register instruksi menyimpan 4-bit opcode dan 4-bit operand (data)

D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul VSM_RingCounter.SCH, modul VSM_Counter16.SCH, VSM_InstructionReg.SCH.

E. PROSEDUR PERCOBAAN

Generator Fase

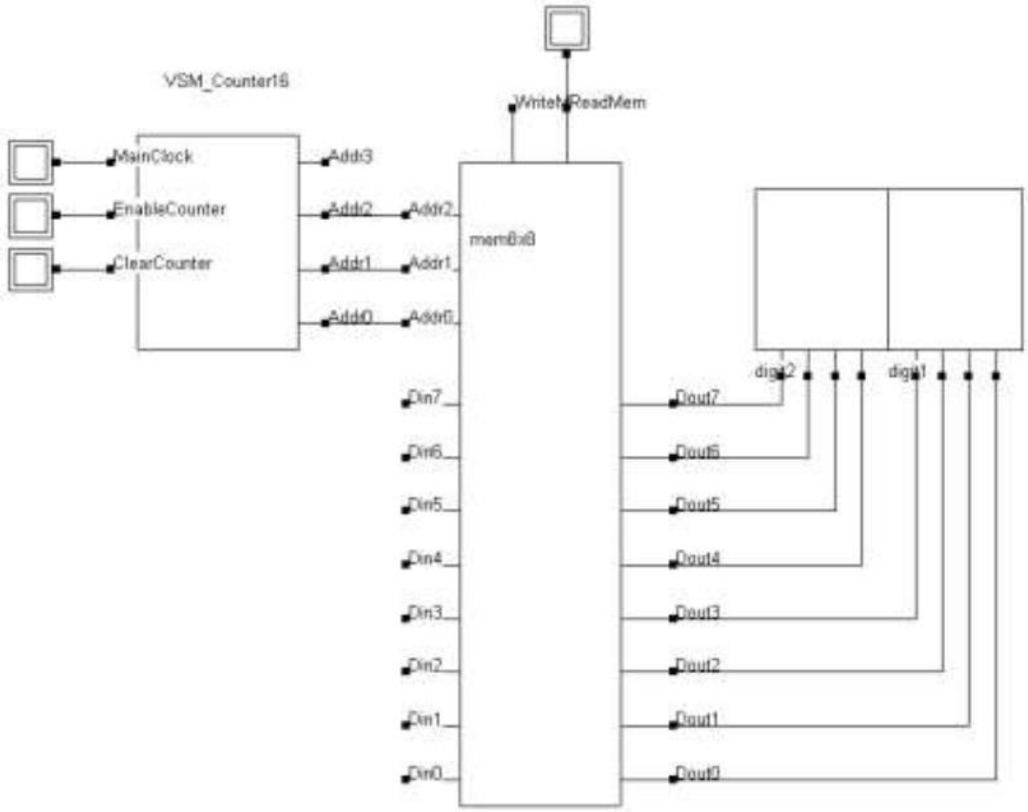
1. Load modul VSM_RingCounter.SCH sehingga diperoleh rangkaian seperti pada gambar 55.
2. Lakukan pengujian terhadap kerja generator fase. Generator fase dapat bekerja dengan baik jika wataknya sama dengan ring counter.
3. Jika modul itu telah dapat bekerja dengan baik, simpanlah ke dalam bentuk simbol dengan nama VSM_RingCounter.SYM.
4. Lakukan pengujian simbol tersebut.

Program Counter

1. Load modul VSM_Counter16.SCH sehingga diperoleh rangkaian seperti pada gambar 59.
2. Lakukan pengujian terhadap kerja Program Counter. Program counter dapat bekerja dengan baik jika wataknya sama dengan counter modulo-16.
3. Jika modul itu telah dapat bekerja dengan baik, simpanlah ke dalam bentuk simbol dengan nama VSM_Counter16.SYM.
4. Lakukan pengujian simbol tersebut.

Memori

1. Susun rangkaian pengujian memori seperti gambar berikut ini dengan menggunakan editor DSCH2. Simpan ke dalam file Mem_Tes.SCH.



Gambar 61. Rangkaian pengujian memori

2. Jalankan DSCH3 dan load file Mem_Tes.SCH.
3. Klik double pada simbol memori dan isilah memori dengan data 1010 0000 (alamat 0), 1011 0001 (alamat 1), 1100 0010 (alamat 2), 1101 0011 (alamat 3), 1110 0100 (alamat 4), 1111 0101 (alamat 5), 0000 0110 (alamat 6), dan 0001 0111 (alamat 7).
4. Lakukan pengujian memori dengan memberikan ReadMem=1, EnableCounter=1, ClearCounter=1.
5. Berikan clock sebanyak 7 kali, apakah alamat memori dibaca secara berurutan dari alamat 0 sampai dengan 7? Apakah hasil bacaan memori sama dengan isi memori?

F. TUGAS AKHIR

1. Apa fungsi memori dan kontroler mikroinstruksi dalam suatu sistem mikrokomputer?
2. Apa perbedaan instruksi dengan mikroinstruksi?

PERCOBAAN X
VSM (VERY SIMPLE MICROPROCESSOR)
BAGIAN VI: UNIT KONTROL DAN RANGKAIAN VSM

A. TUJUAN PERCOBAAN

Melalui percobaan ini mahasiswa diharapkan dapat membangun unit kontrol pada mesin VSM dan menjelaskan cara kerja VSM secara menyeluruh.

B. TUGAS PENDAHULUAN

Kerjakan tugas pendahuluan berikut sebelum melakukan praktikum:

1. Bagaimana Unit Kontrol VSM mengontrol aliran instruksi dalam program VSM??

C. TEORI

Pada praktikum ini akan dibangun mikroprosesor yang berkerja berbasis pada program yang diberikannya (*fully programmable microprocessor*). Praktikum pada percobaan 10 ini melanjutkan percobaan sebelumnya. Langkah pertama yang perlu dilakukan adalah membangun terlebih dahulu rangkaian generator fase yang dapat membangkitkan sinyal fase1, fase2, fase3, dan fase4 setiap terjadinya tepi turun atau negatif dari clock karena mesin VSM ini dalam melaksanakan instruksinya berbasis pada pelaksanaan instruksi 4 fase.

Kontroler Mikroinstruksi

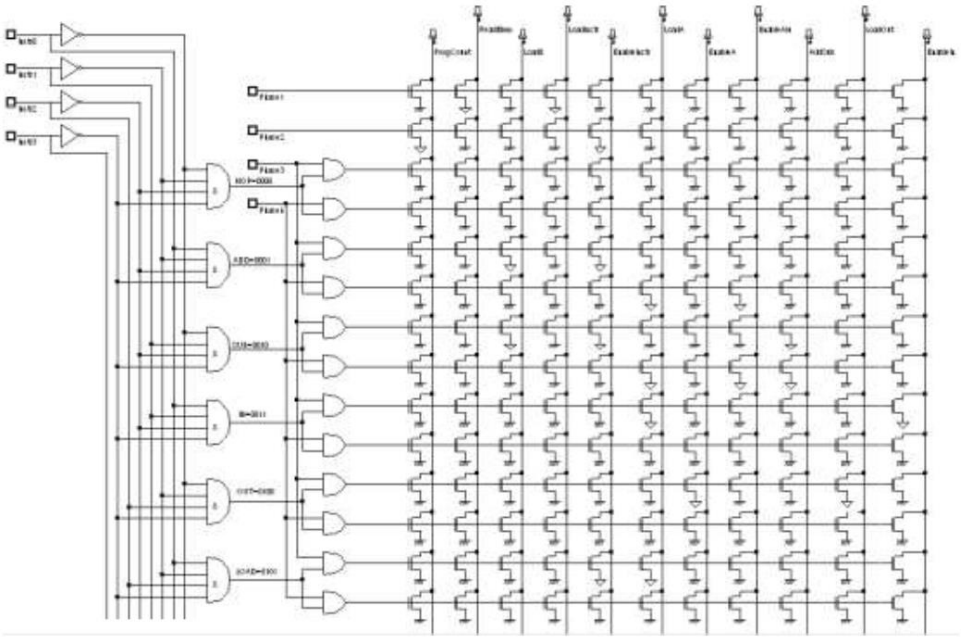
Mikroinstruksi adalah jantung dari mikroprosesor yang membangkitkan sinyal-sinyal pengontrol seperti enable dan latch. Input kontroler mikroinstruksi adalah kode-kode instruksi itu sendiri ditambah informasi fase. Untuk membangun dekoder instruksi yang ada dalam kontroler mikroinstruksi digunakan 6 buah NAND. Keenam gerbang NAND tersebut digunakan untuk mendekode 6 instruksi yakni NOP (0000), ADD (0001), SUB (0010), IN (0011), OUT (0100), dan LDA (0101). Operasi dekoder tersebut dapat dijelaskan sebagai berikut. Jika pada input kontroler mikroinstruksi terdapat instruksi NOP (0000), maka gerbang NAND paling atas outputnya 1 sementara gerbang AND yang lain 0.

Untuk memperoleh rangkaian yang dapat membangkitkan sinyal kontrol yang diperlukan, perlu diidentifikasi terlebih dahulu mikroinstruksi untuk semua instruksi berdasarkan fase pelaksanaannya.

Tabel 13. Mikroinstruksi yang diperlukan pada setiap fase setiap instruksi

Instruksi	Fase 1	Fase 2	Fase 3	Fase 4
NOP (0000)	ReadMem LoadInstr	EnableInstr ProgCount	- -	- -
ADD (0001)	ReadMem LoadInstr	EnableInstr ProgCount	EnableInstr LoadB	EnableAlu LoadA
SUB (0010)	ReadMem LoadInstr	EnableInstr ProgCount	EnableInstr LoadB AddSub	EnableAlu LoadA AddSub
IN (0011)	ReadMem LoadInstr	EnableInstr ProgCount	EnableIn LoadA	- -
OUT (0100)	ReadMem LoadInstr	EnableInstr ProgCount	LoadOut LoadA	- -
LDA (0101)	ReadMem LoadInstr	EnableInstr ProgCount	EnableInstr LoadA	- -

Rancangan dari kontroler mikroinstruksi menggunakan matriks transistor seperti ditunjukkan pada gambar 62. Dari tabel 13 terlihat bahwa pada fase 1 harus dibangkitkan mikroinstruksi ReadMem dan LoadInstr. Untuk menghasilkan kondisi tersebut, terminal drain dari transistor-transistor yang terhubung dengan fase 1 dan ReadMem serta LoadInstr diberi nilai tinggi (+Vcc). Dengan demikian jika terdapat sinyal pada input fase 1, transistor-transistor tersebut akan ON menyebabkan ReadMem dan LoadInstr bernilai tinggi. Dengan menggunakan cara yang sama maka dapat diperoleh rangkaian pengontrol mikroinstruksi seperti pada gambar 61.



Gambar 62. Rangkaian kontroler mikrooperasi yang membangkitkan sinyal enable

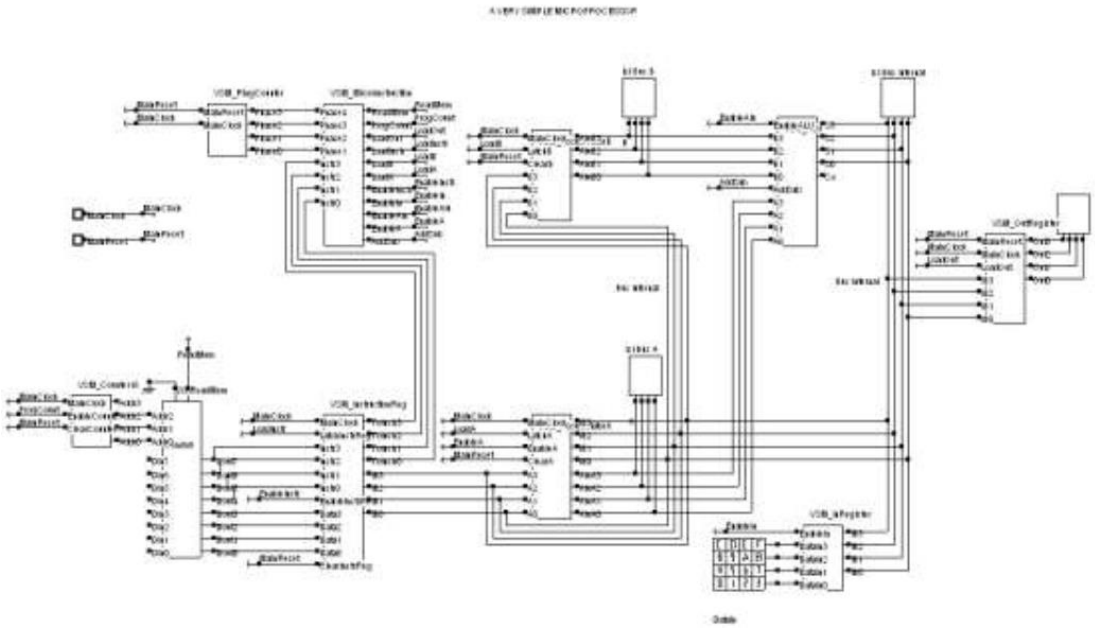
Rangkaian Lengkap VSM

Dengan menggabungkan semua sub rangkaian, dapat diperoleh rangkaian lengkap VSM seperti ditunjukkan pada gambar 63. Sebelum VSM dijalankan perlu diisi terlebih dahulu memori dengan program yang akan dilaksanakan oleh VSM. Contoh program ditunjukkan pada tabel 14.

Tabel 14. Contoh program yang akan disimpan ke dalam memori

Mnemonic	Opcode | Operand Dalam Biner	Opcode | Operand Dalam Heksadesimal
LDA 2	0101 0010	52
ADD 1	0001 0001	11
OUT	0100 0000	40
NOP	0000 0000	00

Perlu diingat pula setiap kali akan menjalankan VSM, berikan sinyal MainReset=1 dan atur agar frekuensi clock tidak terlalu tinggi.



Gambar 63. Rangkaian VSM

D. ALAT-ALAT PERCOBAAN

1. Sebuah komputer/PC
2. *Software* Editor dan Simulator Logika DSCH2
3. Modul VSM_RingCounter.SCH, modul VSM_Counter16.SCH, VSM_InstructionReg.SCH, VSM_Microinstruction.SCH dan modul VSM.SCH.

E. PROSEDUR PERCOBAAN

Kontroler Mikroinstruksi

1. Load file VSM_Microinstruction.SCH sehingga diperoleh rangkaian seperti pada gambar 62.
2. Lakukan pengujian rangkaian ini dalam membangkitkan sinyal kontrol. Apakah rangkaian telah dapat membangkitkan sinyal kontrol sesuai dengan tabel 12?
3. Jika rangkaian telah dapat bekerja dengan baik, simpan dalam bentuk simbol dengan nama VSM_Microinstruction.SYM, kemudian lakukan pengujian terhadap simbol tersebut.

Rangkaian VSM

1. Dengan menggabungkan semua modul yang telah diperoleh/dirancang, susunlah rangkaian VSM lengkap seperti gambar 63. Jika anda ingin memperoleh rangkaian yang sudah jadi, load file VSM.SCH.
2. Isilah terlebih dahulu memori dengan data seperti pada tabel 14.
3. Berikan sinyal MainReset=1, amati kerja VSM! Apakah VSM telah dapat melakukan penjumlahan bilangan 2 dengan 1, dan hasilnya disimpan pada accumulator A serta peraga 7-segmen.
4. Coba gantilah instruksi-instruksi yang ada dalam memori dengan instruksi-instruksi untuk mengurangkan bilangan 3 dengan 2, dan hasilnya ditampilkan pada peraga 7-segmen.
5. Ganti instruksi dalam memori dengan instruksi untuk menambahkan data 4 dari unit input dengan bilangan 3 dan hasilnya ditampilkan ke peraga 7-segmen.

F. TUGAS AKHIR

1. Secara ringkas, jelaskan cara kerja VSM dalam menjumlahkan bilangan 2 dan 1!

DAFTAR PUSTAKA

Malvino, A. P. and Brown J. A. Digital Computer Electronics. Lake Forest: Glencoe Division of Macmillan/McGraw-Hill School Publishing Company.

Mano, M. M. 1992. Computer System Architecture (3rd Edition). Englewood Cliff: Prentice Hall, Inc.

Muchlas. 2005. Rangkaian Digital. Yogyakarta: Gava Media.

Murdocca, M. and Heuring, V. P. 1999. Principles of Computer Architecture. Englewood Cliff: Prentice Hall, Inc.

Sicard, E. 2005. A Very Simple Microprocessor. www.microwind.org

Smith, R. J. and Dorf, R. C. 1992. Circuits, Devices and Systems. New York: John wiley & Sons, Inc.

Tocci, R. J. & Widmer, R. S. 2001. Digital Systems: Principles and Applications, 8th Edition. Englewood Cliff: Prentice Hall, Inc.