PRAKTIKUM 1: PENGENALAN CPU SIMULATOR

Pertemuan ke : 1

Total Alokasi Waktu : 90 menit
- Materi : 15 menit
- Pre-Test : 15 menit
- Praktikum : 45 menit
- Post-Test : 15 menit

Total Bobot Penilaian : 100%
- Pre-Test : 20 %
- Praktik : 30 %
- Post-Test : 50 %

Pemenuhan CPL dan CPMK:

CPL-03	Mampu menerapkan konsep teoritis bidang area Informatika terkait matematika dasar dan ilmu komputer untuk memodelkan masalah dan meningkatkan produktivitas
CPMK-01	Mampu menjelaskan diagram blok komputer dan dasar-dasar sistem operasi meliputi sejarah perkembangan dan tujuannya, keterkaitan sumber daya dan macam-macam layanan dalam sistem operasi

1.1. DESKRIPSI CAPAIAN PEMBELAJARAN

Setelah mengikuti praktikum ini mahasiswa diharapkan mampu:
1. Menjelaskan sistem operasi komputer serta perkembangan dan fungsi-fungsinya
2. Menerapkan instruksi untuk memindahkan data ke register, membandingkan isi register, memasukkan data ke stack, mengambil data dari stack, melompat ke lokasi alamat, menambahkan nilai dalam register.
3. Menjelaskan fungsi-fungsi dari register khusus CPU antara lain register PC, SR, dan SP.

1.2. INDIKATOR KETERCAPAIAN PEMBELAJARAN

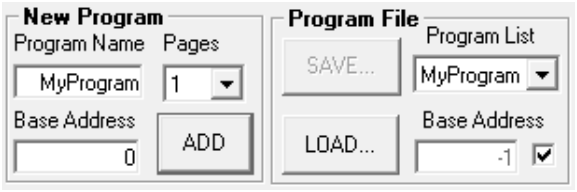
Indikator ketercapaian diukur dengan:

CPL-03	CPMK-01	Kemampuan mahasiswa dalam menerapkan instruksi untuk memindahkan data ke register, membandingkan isi register, memasukkan data ke stack, mengambil data dari stack, melompat ke lokasi alamat, menambahkan nilai dalam register

1.3. TEORI PENDUKUNG

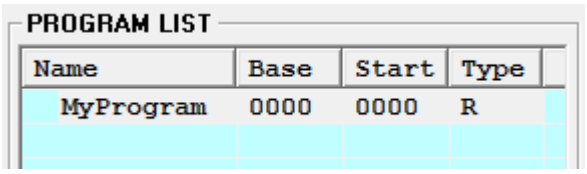
Di bawah ini adalah deskripsi dari 4 tahap utama:

1. Membuat program CPU yang berisi instruksi-instruksi



Gambar 1. 1 New Program Frame

Masukkan nama program pada **Program Name** text box, contoh MyProgram. Masukkan nomor pada **Base Address** text box (saran = gunakan 0 untuk kasus ini). Kemudian klik tombol **ADD**. Nama program akan muncul pada frame LIST PROGRAM (Gambar 1.2).



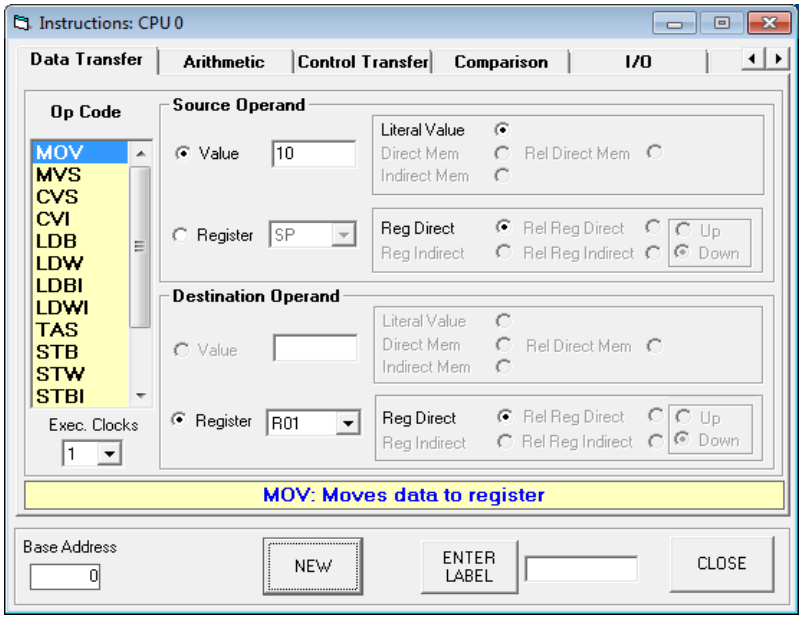
Gambar 1. 2 Program list frame

2. Adding CPU instructions in the program

Gambar 1. 3 Program instructions frame

Dalam frame **PROGRAM INSTRUCTION** tombol **ADD NEW** aktif. Klik tombol tersebut untuk melihat CPU instruction kamu dapat memilih untuk program yang dibuat di atas (Gambar 1.3).

Memilih CPU instructions untuk memasuki program secara manual:

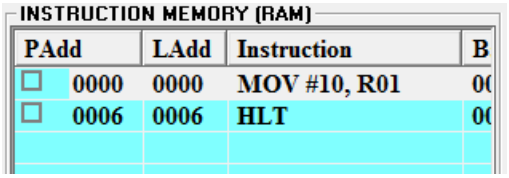


Gambar 1. 4 Selecting CPU instructions

Pada window **Instruction** yang diperlihatkan di atas, kamu dapat memiih beberapa CPU instruction yang tersedia untuk programmu. Instructions yang terhubung dikategorikan dalam beberapa kelompok. Kamu dapat memilih sebuah kelompok dengan mengklik pada tab grup. Pilih instruction yang diinginkan dari list di bawah **OP CODE.** Jika instruction yang dipilih membutuhkan operand(s) maka tipe yang tersedia akan aktif di bawah **SOURCE OPERAND** dan frame **DESTINATION OPERAND.** Selanjutnya, klik tombol **NEW** untuk menamba instuction. Kamu dapat menambah instruction ganda tanpa menutup window ini.

Instruction akan muncul pada area memori CPU program seperti yang diperlihatkan pada Gambar 5. Simulator CPU membutuhkan membutuh instruction terakhir pada program yaitu **HLT** instruction (intructio ini memberhentikan proses simulasi. Seperti yang terdapat pada Gambar 1.5 Program sekarang dapat dijalankan.

CPU instructions in program:



Gambar 1. 5 CPU program memory

Setiap entry pada view **INSTRUCTION MEMORY** terdapat informasi sebagai berikut:

Padd(Physicial Address), **Ladd**(Logical Adress) dan instruction. Informasi lainnya juga tersedia tapi tidak relevan pada tahap ini.

Editing the program instructions:

Gambar 1. 6 Program edit functions

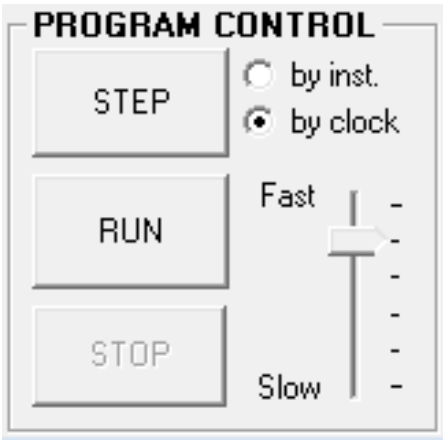
Sekali instructions program dimasukkan. Mereka dapat diedit. Untuk melakukannya, pilih instruction yang diinginkan dan gunakan satu dari fungsi edit (**EDIT, DELETE, MOVE UP, MOVE DOWN)** diperlihatkan pada Gambar 1.6 untuk mengedit instruction yang dipilih. Gunakan tombol **INSERT ABOVE...** dan **INSERT BELOW...** untuk memasukkan instruction baru diatas(above) atau di bawah (below) instruction yang dipilih.

Menghapus program:

Gambar 1. 7 Program removal

Program CPU dapat dihapus dengan mengklik tombol **REMOVE PROGRAM** diperlihatkan pada Gambar 1.7 Sekali program dihapus, maka program tersebut tidak akan ada lagi dan hilang. Bagaimanapun kamu dapat menyimpan program tersebut sehingga kamu dapat membuka kembali program tersebut nanti (Gambar 1.12)

3. Running the program

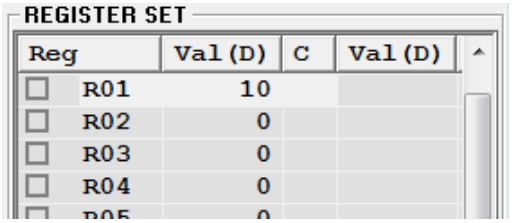


Gambar 1. 8 Program control

Program CPU dapat dijalankan dalam 2 cara yang berbeda : 1) instruction dengan instruction, 2) otomatis dalam sekali jalan. Untuk menjalankan instruction yang dipilih dengan sendirinya, pertama pilih program seperti Gambar 5 dan double klik pada program tersebut. alternatifnya, kamu dapat mengklik tombol **STEP** yang diperlihatkan pada Gambar 8 (pastikan pilihan **by inst.** Dipilih). Gunakan tombol **STOP** untuk memberhentikan program yang berjalan. Gunakan **slider control** untuk mempercepat atau memperlambat program yang berjalan

4. Observing and controlling the simulations

Mengamati/mengubah register CPU



Gambar 1. 9 Register Set

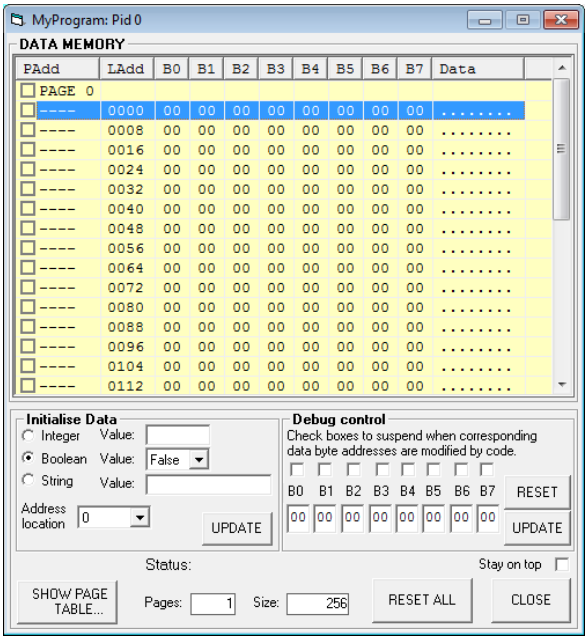
Sebuah instruction yang menulis atau membaca akses register pada frame **REGISTER SET** dan register yang diakses disorot. Frame ini menunjukkan register-register dan nilai mereka. Klik pada kolom **Val** untuk mengubah nilai dari desimal **(D)** ke format hex **(H)** dan sebaliknya.

Note: untuk mengubah nilai register secara manual, pertama pilih dan masukkan nilai pada field **Reg Value** dan klik pada tombol **CHANGE** (Gambar 1.9).

Observing/altering the program data:

Instruction CPU yang mengakses bagian dari memori yang mengandung data dapat menulis atau membaca data yang diakses. Informasi ini tersedia pada window halaman memori pada Gambar 1.10. Klik tombol **SHOW PROGRAM DATA MEMORY...** yang diperilihatkan pada Gambar 7 diatas. Pada window ini juga dapat mengedit isi dari data.

Kolom **Ladd** menunjukkan starting address pada setiap garis di display. Setiap garis dari display merepresentasikan 8byte dari informasi, jadi nomor **Ladd** terurut dari kecil hingga 8 dari setiap garis di bawah display. Kolom **B0** ke **b7** terdapat bytes 0 hingga 7. Kolom **Data** menunjukkan karakter yang dapat diperlihatkan sesuai dengan 8 bytes. Bytes tersebut berhubungan ke character yang tidak dapat diperlihatkan yang ditunjukkan sebagai dots. Data byes hanya dapat diperlihatkan dalam format hex.

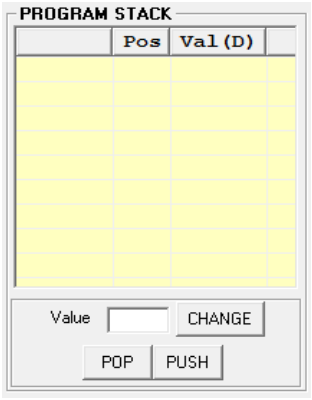


Gambar 1. 10 Data memory page

Untuk mengubah nilai dari bytes manapun pertama pilih garis yang mengandung bytes. Kemudia gunakan informasi pada frame **Initialize Data** untuk memodifikasi nilai dari bytes pada garis yang dipilih sebagai format **Integer, Boolean atau String.** Kamu harus mengklik tombol **UPDATE** untuk mengubah.

Cek check box **Stay on top** untuk memastikan window selalu berada di atas window lainnya ketika masih memperbolehkan mengakses windows di bawahnya.

(melihat/mengubah stack program):

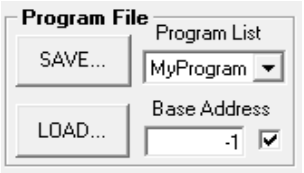


Programs menjadikan **PROGRAM STACK** untuk menyimpan informasi penting sementara seperti **subroutine return addressess** dan **subroutine parameters** dan informasi relevan lainnya. Terdapat instruction yang dapat mendorong **(PSH)** data di atas stack dan dapat memunculkan data (**POP)** data dari atas stack ke register.

Kamu dapat mendorong dan memunculkan data secara manual dengan mengklik tombol **PUSH dan POP.** Kamu juga dapat memodifikasi entry stack dengan memilihnya, memasukkan nilai baru pada text box nilai dan mengklik **CHANGE**

Gambar 1. 11 Program stack frame

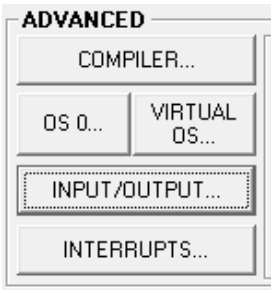
Saving and loading the CPU programs:



Untuk menyimpan program, pilih program dengan mencari program dari down list dan klik tombol **SAVE...** untuk memuat program yang tersimpan klik tombol **LOAD**

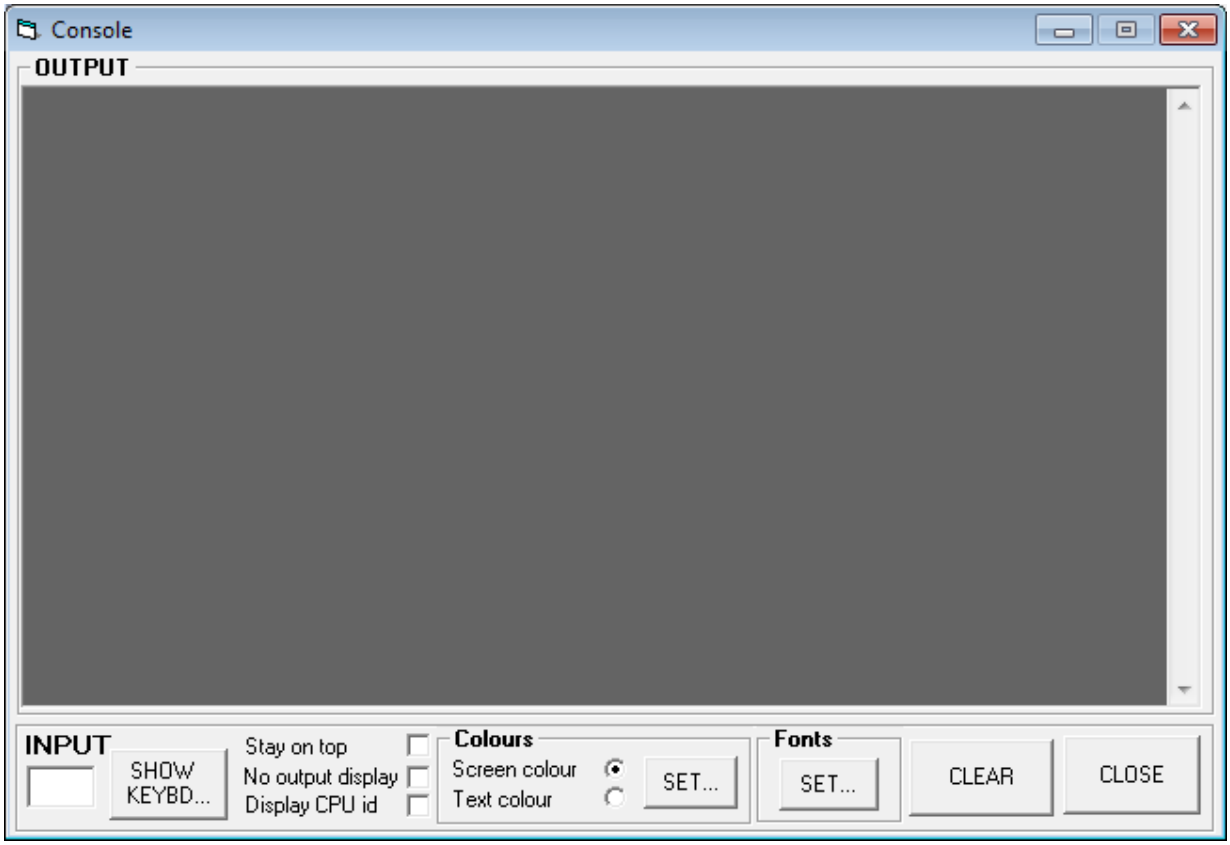
Gambar 1. 12 Saving and loading programs

Observing the displayed information:



Gambar 1. 13 Console button

Program dapat memperlihatkan informasi dan menerima data dari konsol yang disimulasi. Instruction **OUT** digunakan untuk memperlihatkan informasi dan instruction **IN** digunakan untuk menerima input dari konsol. Untuk menunjukkan konsol klik tombol **INPUT/OUTPUT...** ditunjukkan pada Gambar 1.13. Konsol window ditunjukkan pada Gambar 1.14.



Gambar 1. 14 Console button

Cek check box **Stay on top** untuk memastikan window selallu berada di atas windows lainya ketika masih memperbolehkan akses windows di bawahnya. Klik pada **SHOW KEYBD...** untuk menunjukkan keyboard kecil virtual untuk menginput data pada Gambar 1.15.

Gambar 1. 15 Virtual keyboard

Cek check box **Lower Case** untuk menginput karakter kecil. Keyboard ini adalah bagian dari keyboard standard. Kamu juga dapat menginput dengan menulis dapat text box **INPUT** yang ditunjukkan pada Gambar 1.14

Tabel 1. 1 CPU Simulator Instruction

Instruction	Description
Data transfer instructions	
MOV	Move data to register; move register to register e.g. **MOV #2, R01** moves number 2 into register R01 **MOV R01, R03** moves contents of register R01 into register R03
LDB	Load a byte from memory to register
LDW	Load a word (2 bytes) from memory to register
STB	Store a byte from register to memory
STW	Store a word (2 bytes) from register to memory
PSH	Push data to top of hardware stack (TOS); push register to TOS e.g. **PSH #6** pushes number 6 on top of the stack **PSH R03** pushes the contents of register R03 on top of the stack
POP	Pop data from top of hardware stack to register e.g. **POP R05** pops contents of top of stack into register R05
Arithmetic instructions	
ADD	Add number to register; add register to register e.g.

	ADD #3, R02 adds number 3 to contents of register R02 and stores the result in register R02. **ADD R00, R01** adds contents of register R00 to contents of register R01 and stores the result in register R01.
SUB	Subtract number from register; subtract register from register
MUL	Multiply number with register; multiply register with register
DIV	Divide number with register; divide register with register
Control transfer instructions	
JMP	Jump to instruction address unconditionally e.g. **JMP 100** unconditionally jumps to address location 100
JLT	Jump to instruction address if less than (after last comparison)
JGT	Jump to instruction address if greater than (after last comparison)
JEQ	Jump to instruction address if equal (after last comparison) e.g. **JEQ 200** jumps to address location 200 if the previous comparison instruction result indicates that the two numbers are equal.
JNE	Jump to instruction address if not equal (after last comparison)
CAL	Jump to subroutine address
RET	Return from subroutine
SWI	Software interrupt (used to request OS help)
HLT	Halt simulation
Comparison instruction	
CMP	Compare number with register; compare register with register e.g. **CMP #5, R02** compare number 5 with the contents of register R02 **CMP R01, R03** compare the contents of registers R01 and R03 Note: If R01 = R03 then the status flag Z will be set

	If R03 > R01 then none of the status flags will be set If R01 > R03 then the status flag N will be set
Input, output instructions	
IN	Get input data (if available) from an external IO device
OUT	Output data to an external IO device

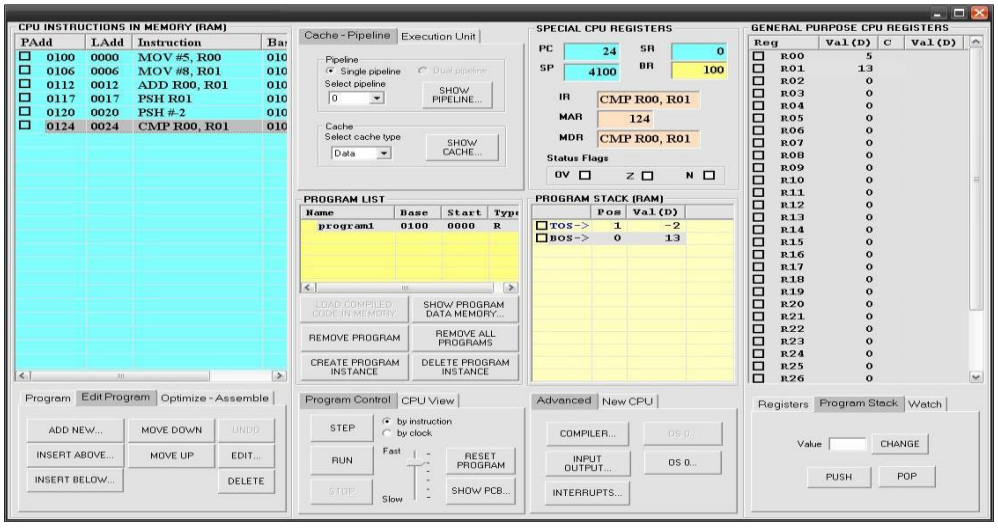
Model Pemrograman

Model pemrograman arsitektur komputer mendefinisikan komponen-komponen arsitektur pada tingkat bawah (low level architectural components) yang mencakup:

a. Set instruksi prosesor
b. Register-register
c. Jenis-jenis pengalamatan instruksi dan data
d. *Interrupts* dan *exceptions*

Pada model pemrograman ini terdapat interaksi antar komponen diatas. Ini merupakan model pemrograman tingkat rendah (low-level) yang memungkinkan suatu komputasi terprogram.

Perangkat Simulator

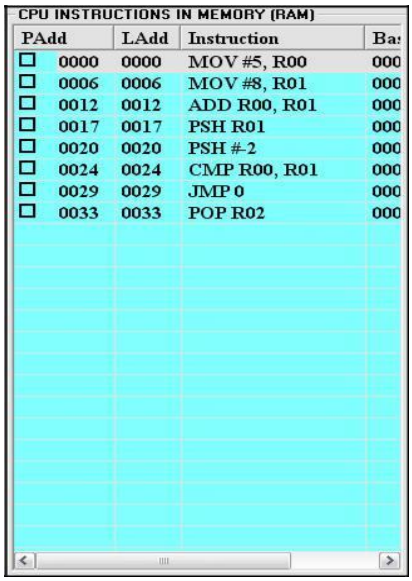


Gambar 1. 16 Jendela utama simulator

Jendela utama terdiri dari beberapa tampilan, yang merepresentasikan beberapa bagian fungsional prosesor yang disimulasikan, yaitu:

a. Instruction memory (RAM)
b. Special registers
c. Register set
d. Program stack

Tampilan *instruction memory*

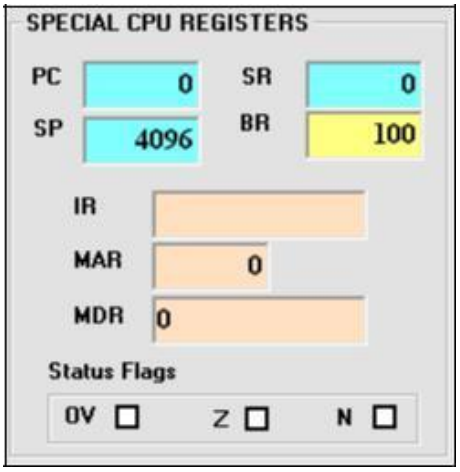


Tampilan berisi instruksi-instruksi dari program. Instruksi ditampilkan dalam urutan instruksi mnemonik low-level (format assembler), tidak dalam bentuk kode biner. Hal ini bertujuan untuk memperjelas dan membuat kode lebih mudah dibaca.

Tiap instruksi mempunyai dua alamat: alamat fisik (Padd) dan alamat logika (Ladd). Tampilan ini juga menyajikan alamat dasar (Base) terhadap tiap instruksi. Urutan instruksi pada program yang sama akan mempunyai alamat dasar yang sama.

Gambar 1. 17 Tampilan instruction view

Tampilan *Special Registers*



Gambar 1.18 Tampilan Special Register

Gambar 1.18 menyajikan register-register dengan fungsi-fungsi khusus yang telah ditentukan:

PC: Program Counter, berisi alamat instruksi berikutnya yang akan dieksekusi.

IR: Instruction Register, berisi instruksi yang sedang dieksekusi saat ini.

SR: Status Register, berisi informasi yang memberikan hasil dari instruksi yang dieksekusi terakhir.

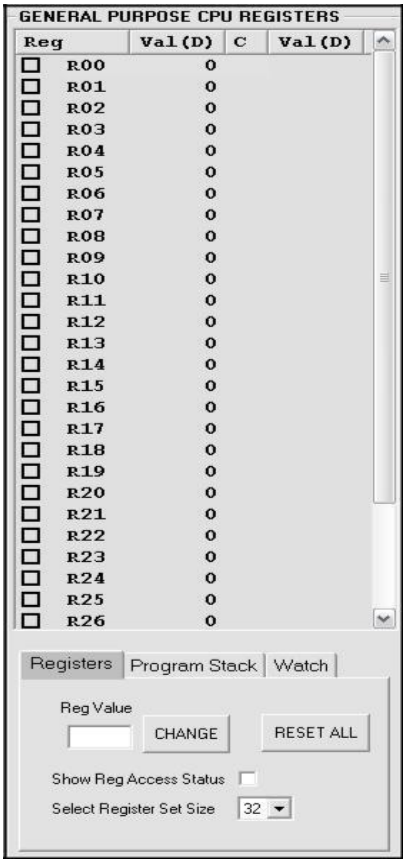
SP: Stack Pointer, register menunjuk ke nilai yang berada pada bagian atas stack.

BR: Base Register, berisi alamat dasar saat ini.

MAR: Memory Address Register, berisi alamat memori yang sedang diakses.

Status Bits: OV: Overflow; **Z**: Zero; **N**: Negative

Tampilan *Register Set*



Gambar 1.19 Tampilan Register Set

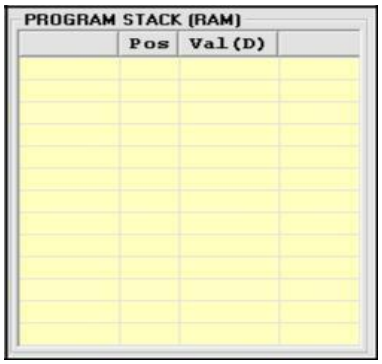
Tampilan register set menunjukkan isi dari semua register-register tujuan umum (general-purpose), yang digunakan untuk menampung nilai sementara ketika instruksi program dieksekusi.

Arsitektur ini mendukung 8 hingga 64 register. Register-register ini sering digunakan untuk menyimpan nilai variabel program pada bahasa tingkat tinggi.

Tidak semua arsitektur memiliki register sebanyak ini. Beberapa lebih banyak (misalnya 128 register) dan beberapa lainnya lebih sedikit (misalnya 8 register). Pada semua kasus, register-register ini bekerja untuk tujuan yang sama.

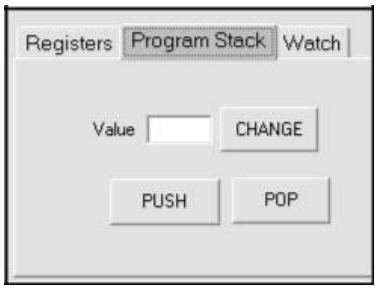
Bagian ini menampilkan nama tiap register (Reg), nilai (Val), dan beberapa informasi lainnya untuk keperluan debug program. Kita juga dapat melakukan reset (RESET) atau mengisikan nilai register (CHANGE) secara manual.

Tampilan *Program Stack*



Gambar 1.20 Tampilan Program Stack

Program Stack menampung nilai sementara ketika instruksi dieksekusi. Stack menggunakan struktur data LIFO (last-In-First-Out). Stack sering digunakan untuk efisiensi *interrupt handling* dan *sub-routine call*.



Instruksi PUSH dan POP digunakan untuk menyimpan dan mengambil nilai pada bagian teratas stack.

1.4. HARDWARE DAN SOFTWARE

Hardware dan software yang digunakan dalam praktikum ini yaitu:

1. Komputer.
2. CPU Simulator

1.5. PRE-TEST

Jawablah pertanyaan berikut (**Total Skor: 100):**

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-01	Jelaskan fungsi dari CPU Simulator!	20
2.	CPL-03	CPMK-01	Sebutkan dan jelaskan instruksi dari Special Register	40
3.	CPL-03	CPMK-01	Jelaskan cara menempatkan instruksi ke instruction memory!	40

1.6. LANGKAH PRAKTIKUM

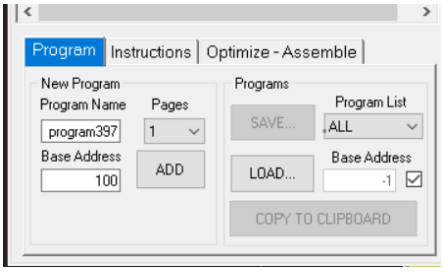
Aturan Penilaian (Total Skor: 100):

No	CPL	CPMK	Pertanyaan	Dokumen Pendukung	Skor
1.	CPL-03	CPMK-01	Selesaikan langkah praktikum A-B	Hasil praktikum langkah A-B	100

Langkah-Langkah Praktikum:

A. Persiapan Praktikum

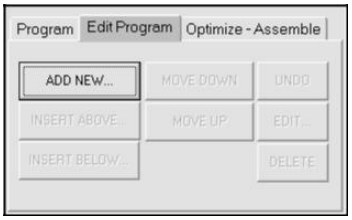
Pertama-tama perlu menempatkan sejumlah instruksi pada tampilan instruction memory (menggambarkan RAM pada mesin yang nyata) sebelum melakukan eksekusi instruksi. Berikut adalah cara menempatkan instruksi ke instruction memory:



Gambar 1.21 Tampilan Program Instruction

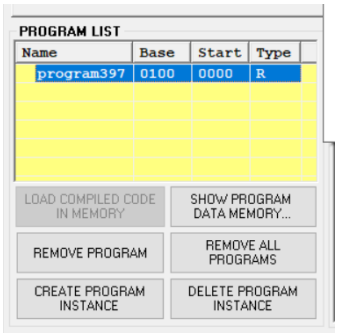
Pada tampilan program instruction, masukkan nama program, dan juga alamat dasar (untuk praktikum ini beri alamat dasar nilai 100). Klik tombol ADD. Maka program baru dengan namanya akan dimasukkan ke tampilan Program List seperti yang disajikan gambar 1.22. Gunakan tombol SAVE.../ LOAD... untuk menyimpan dan mengambil instruksi dari file.

Gunakan tombol REMOVE PROGRAM untuk menyingkirkan program yang disorot. Gunakan tombol REMOVE ALL PROGRAMS untuk menyingkirkan semua program yang ada di daftar. Sebagai catatan, ketika



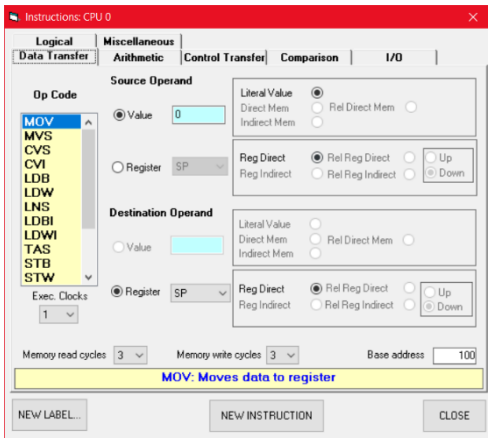
program disingkirkan maka instruksi-instruksi yang terkait dengannya juga akan dihapus dari tampilan instruction memory.

B. Langkah Praktikum



Gambar 1.22 Tampilan Program List

Untuk memasukkan instruksi ke CPU, klik tombol ADD NEW... yang akan menampilkan jendela Instructions: CPU0.



Gunakan jendela ini untuk memasukkan instruksi. Pada **Appendix** tersedia beberapa instruksi yang dikenali oleh simulator ini dan terdapat pula contoh penggunaannya.

a. Transfer data

1. Buatlah instruksi yang memindahkan (move) angka 5 ke register R00.
2. Eksekusi instruksi diatas (dengan klik dua kali pada tampilan instruction memory).
3. Buatlah instruksi yang memindahkan angka 8 ke register R01 dan eksekusilah.
4. Amati isi R00 dan R01 pada tampilan Register Set.

b. Aritmatika

6. Buatlah suatu instruksi yang menambahkan (add) isi R00 dan R01.
7. Eksekusilah.
8. Amati dimana hasil penjumlahan tersebut disimpan.

c. Stack Pointer (SP)

9. Buatlah instruksi yang menaruh hasil diatas pada program stack, kemudian eksekusilah.
10. Buatlah instruksi untuk menaruh (push) angka -2 pada stack teratas dan eksekusilah.
11. Amati nilai register SP.

d. Pembanding

12. Buatlah instruksi untuk membandingkan nilai register R00 dan R01. Eksekusilah.
13. Amati nilai register SR.
14. Amati bit status OV/Z/N pada status register.
15. Analisa status register tersebut.

e. Stack Pointer (SP)

16. Buatlah instruksi untuk mengambil (pop) nilai teratas dari program stack ke register R02. Eksekusilah.
17. Amati nilai pada register SP.
18. Buatlah suatu instruksi untuk mengambil nilai teratas dari program stack ke register R03. Eksekusilah.
19. Amati nilai pada register SP.
20. Eksekusi lagi instruksi yang terakhir. Apa yang terjadi? Jelaskan.

1.7. POST TEST

Jawablah pertanyaan berikut (**Total Skor: 100):**

No	CPL	CPMK	Pertanyaan	Skor
1.	CPL-03	CPMK-01	Jelaskan langkah - langkah membuat program sederhana yaitu instruksi menambahkan isi R00 dan R01 dengan hasil akhir 10, wajib disertai screenshot.	25
2	CPL-03	CPMK-01	Jelaskan langkah - langkah membuat instruksi memindahkan angka 13 ke register R03, wajib disertai screenshot.	25
3.	CPL-03	CPMK-01	Jelaskan setiap langkah praktikum di atas menggunakan bahasa anda sendiri-sendiri dan berikan kesimpulan!	50

1.8. HASIL CAPAIAN PRAKTIKUM

Diisi oleh asisten setelah semua assessment dinilai.

No	Bentuk Assessment	CPL	CPMK	Bobot	Skor (0-100)	Nilai Akhir (Bobot x Skor)
1.	Pre-Test	CPL-03	CPMK-01	20%		
2.	Praktik	CPL-03	CPMK-01	30%		
3.	Post-Test	CPL-03	CPMK-01	50%		
					Total Nilai	

LEMBAR JAWABAN PRE-TEST DAN POST-TEST PRAKTIKUM SISTEM OPERASI

Nama : **NIM :**	**Asisten:** **Paraf Asisten:**	**Tanggal:** **Nilai:**