MODUL PRAKTIKUM SISTEM OPERASI

PRAKTIKUM I

MODEL PEMROGRAMAN 1

A. Tujuan

Pada akhir praktikum ini, peserta dapat:

1. Memahami komponen arsitektur komputer tingkat bawah.
2. Menggunakan simulator untuk membuat dan mengeksekusi instruksi dasar CPU.
3. Membuat 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.
4. Menjelaskan fungsi-fungsi dari register khusus CPU antara lain register PC, SR, dan SP.

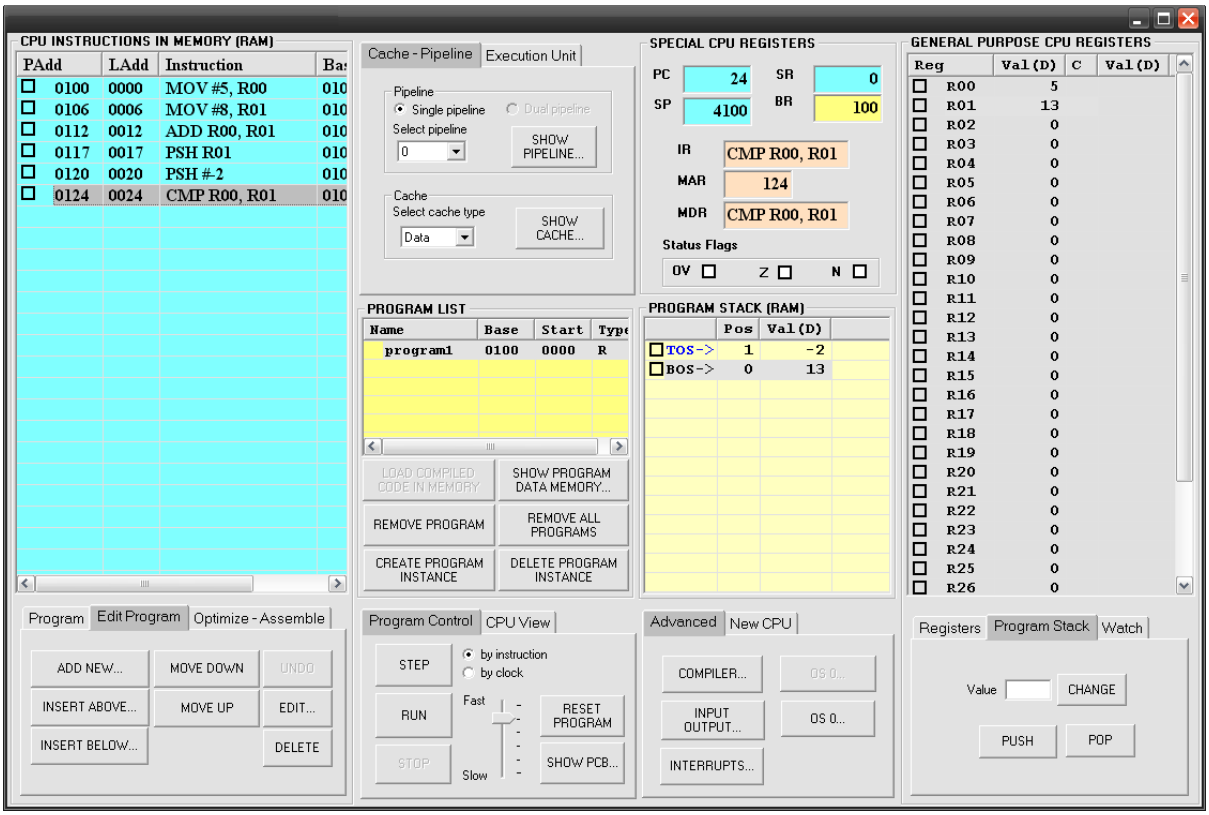
B. Dasar Teori

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

- Set instruksi prosesor
- Register-register
- Jenis-jenis pengalamatan instruksi dan data
- *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.

C. Perangkat Simulator

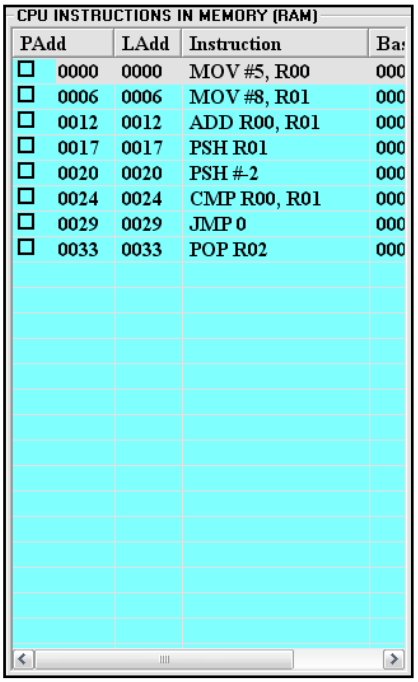


Gambar 1. Jendela utama simulator

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

- Instruction memory (RAM)
- Special registers
- Register set
- Program stack

Tampilan *instruction memory*

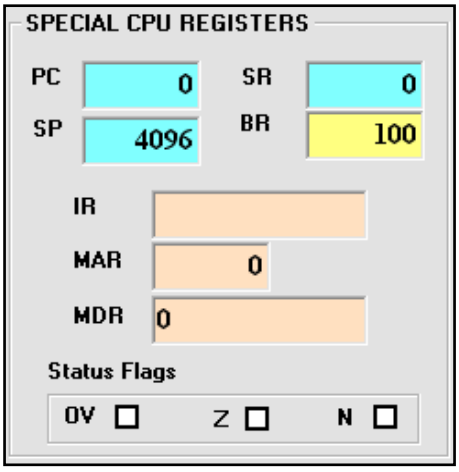


Tampilan ini berisi instruksi-instruksi dari program. Instruksi ditampilkan dalam urutan instruksi mnemonik low-level (format assembler), tidak dalam bentuk kode biner. Hal ini bertujuan untuk memperjelas dan membuat kode lebih mudah dibaca.

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 2. Tampilan *instruction view*

Tampilan *Special Registers*



Gambar 3. Tampilan Special Register

Tampilan ini 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.

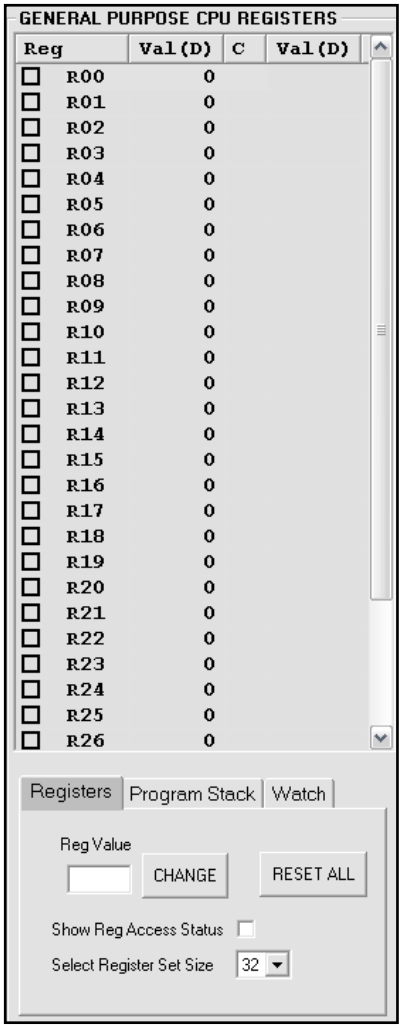
SP : Stack Pointer, register menunjuk ke nilai yang berada pada bagian atas stack.

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*



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

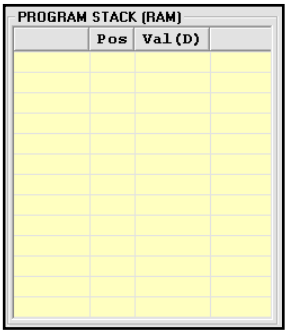
Arsitektur ini mendukung 8 hingga 64 register. Register-register ini sering digunakan untuk menyimpan nilai variabel program pada bahasa tingkat tinggi.

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.

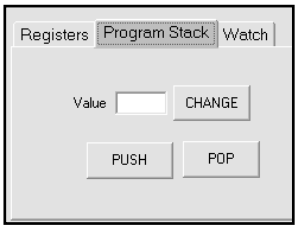
Gambar 4. Tampilan Register Set

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*.

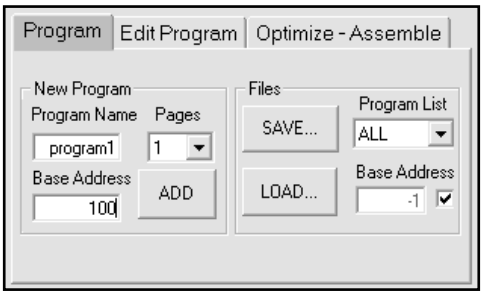
Gambar 5. Tampilan Program Stack



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

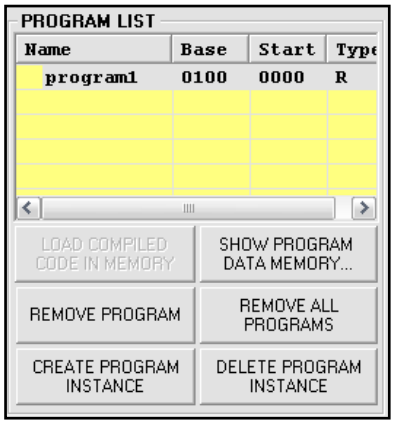
D. Persiapan Praktikum

Pertama-tama kita 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 :



Pada tampilan program instruction, masukkan nama program (misal: program1), 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 7. Gunakan tombol SAVE.../ LOAD... untuk menyimpan dan mengambil instruksi dari file.

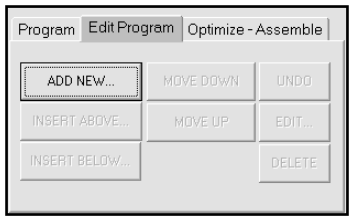
Gambar 6. Tampilan Program Instructions.



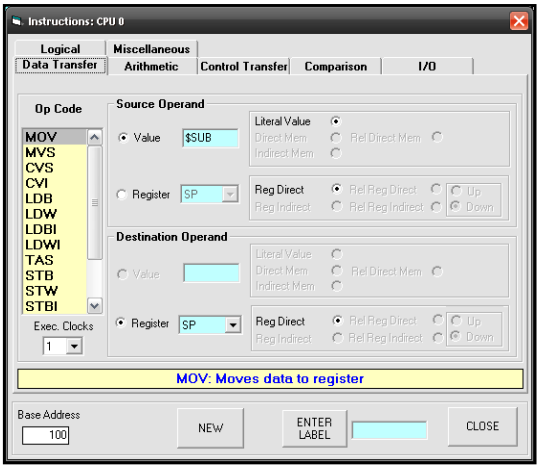
Gunakan tombol REMOVE PROGRAM untuk menyingkirkan program yang disorot. Gunakan tombol REMOVE ALL PROGRAMS untuk menyingkirkan semua program yang ada di daftar. Sebagai catatan, ketika program disingkirkan maka instruksi-instruksi yang terkait dengannya juga akan dihapus dari tampilan instruction memory.

Gambar 7. Tampilan Program List

E. Langkah-langkah Praktikum



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.
4. Eksekusilah.
5. 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.
13. Eksekusilah.
14. Amati nilai register SR.
15. Amati bit status OV/Z/N pada status register.

e. Program Counter (PC)

16. Buatlah instruksi untuk melakukan unconditionally jump ke instruksi yang pertama.
17. Eksekusilah.
18. Amati nilai register PC. Apa yang ditunjuk olehnya?

f. Alamat fisik dan alamat logika

19. Amati nilai pada kolom Padd dan Ladd. Apa arti nilai-nilai yang ditunjukkan oleh keduanya? Apa perbedaannya?
20. Apa perbedaan antara nilai Ladd pada instruksi yang pertama dengan nilai LAdd pada instruksi yang kedua? Apa arti nilai tersebut?

g. Stack Pointer (SP)

21. Buatlah instruksi untuk mengambil (pop) nilai teratas dari program stack ke register R02.
22. Eksekusilah.
23. Amati nilai pada register SP.
24. Buatlah suatu instruksi untuk mengambil nilai teratas dari program stack ke register R03.
25. Eksekusilah.
26. Amati nilai pada register SP.
27. Eksekusi lagi instruksi yang terakhir. Apa yang terjadi? Jelaskan.

h. Status Register (SR)

28. Buatlah suatu instruksi pembanding (compare) yang membandingkan nilai dalam register R04 dan R05.
29. Sisipkan secara manual dua nilai yang sama pada register R04 dan R05.
30. Eksekusi instruksi pembanding pada langkah 28 diatas.
31. Manakah diantaras status flag OV/Z/N yang dalam kondisi set? Mengapa?
32. Sisipkan secara manual nilai dalam register R05 yang lebih besar dari register R04.
33. Eksekusilah instruksi pembanding pada langkah 28 diatas.
34. Manakah diantaras status flag OV/Z/N yang dalam kondisi set? Mengapa?
35. Sisipkan secara manual nilai dalam register R04 yang lebih besar dari register R05.
36. Eksekusilah instruksi pembanding pada langkah 28 diatas.
37. Manakah diantaras status flag OV/Z/N yang dalam kondisi set? Mengapa?

i. Jump if...

38. Buatlah instruksi yang akan melompat (jump) ke instruksi yang pertama jika nilai dalam register R04 dan R05 sama.
39. Ujilah instruksi ini dengan mengisikan nilai yang sesuai pada register R04 dan register R05 kemudian eksekusilah instruksi pembanding, lanjutkan dengan mengeksekusi instruksi lompat.
40. Simpan instruksi dalam Instruction Memory ke file dengan klik pada tombol SAVE.

----- end -----