Tugas Besar II
EL2010 – Organisasi dan Arsitektur Komputer
Semester II Tahun 2011/2012

Assembly Interpreter

Pada Tugas Besar 2 kali ini, setiap kelompok diminta untuk membuat sebuah interpreter yang dalam praktikum ini kelompok membuat interpreter untuk Assembly. Interpreter merupakan perangkat lunak yang berfungsi melakukan eksekusi sejumlah instruksi yang ditulis dalam suatu bahasa pemrograman. Interpreter secara umum menerima file yang akan langsung dieksekusi tanpa membuat *executable* file terlebih dahulu.

Pada tugas besar kali ini, interpreter yang dibuat hanya berupa *interpreter* sederhana yang hanya melakukan interpreter terhadap file assembly tersebut untuk kemudian langsung dieksekusi tanpa harus dibuat *executable* filenya ataupun *object* file.

Spesifikasi

Untuk pengerjaan tugas besar 2 kali ini, bahasa yang diperbolehkan untuk mengerjakan tugas adalah **C/C++.**

Dalam pengerjaan tugas besar kali ini, masukan berupa nama file pada argumen ketika menjalankan program (lihat contoh di bawah). Setiap kali eksekusi file yang menjadi argumen di-*load* ke dalam program dan dieksekusi. File masukkan pengguna berupa **file assembly (.asm)** dan apabila yang masukkan file bukan berekstensi **.asm**, maka program hanya menampilkan pesan kesalahan dan keluar.

Contoh :

Nama program adalah **myasmc**, ketika eksekusi pada command prompt adalah

```
C:> myasmc coba.asm (file dieksekusi)
…
… {eksekusi file coba.asm}
…
End Program
```

```
C:> myasmc coba.pas (gagal eksekusi)
File Salah
End Program
```

Format isi dari file assembly masukan seperti yang terlihat pada contoh di bawah ini. Dalam file masukan ini, label tidak boleh berada pada satu baris dengan command**. Label diakhiri dengan ':' dan tidak boleh ada spasi di tengah label**. Komentar diperbolehkan dalam file.

Contoh:

label →

```
Proses:
MOV AH,02h
MOV DL,'A'
MOV CX,10h
```

command

©Laboratorium Sistem Tersebar 2012

label →

```
;proses pengulangan
Ulang:
INT 21h
INC DL
LOOP Ulang
INT 20h
```

command

Register-register yang harus diimplementasikan antara lain

a. AX, yang dibagi menjadi dua bagian, yaitu AH dan AL yang masing-masing berukuran 8 bit dengan AH merupakan 8 bit MSB dari AX dan AL adalah 8 bit LSB dari AX.
b. BX, yang dibagi menjadi dua bagian, yaitu BH dan BL yang masing-masing berukuran 8 bit dengan BH merupakan 8 bit MSB dari BX dan BL adalah 8 bit LSB dari BX.
c. CX, yang dibagi menjadi dua bagian, yaitu CH dan CL yang masing-masing berukuran 8 bit dengan CH merupakan 8 bit MSB dari CX dan CL adalah 8 bit LSB dari CX.
d. DX, yang dibagi menjadi dua bagian, yaitu DH dan DL yang masing-masing berukuran 8 bit dengan DH merupakan 8 bit MSB dari DX dan DL adalah 8 bit LSB dari DX.

Command-command assembly yang harus dibuat adalah:

1. INT <hexadecimal>
   Command INT adalah perintah untuk melakukan *interrupt*. Interrupt disertai dengan nomor interupsi yang akan dilakukan. Beberapa interrupt dapat terdiri dari beberapa fungsi yang berbeda tergantung pada service yang digunakan. Nilai dari service disimpan pada sebuah register sebelum interrupt dipanggil. Berikut adalah daftar interrupt yang perlu diimplementasikan adalah:

   1. Interrupt 16H servis 01H (membaca karakter ASCII)
   2. Interrupt 20H (keluar dari aplikasi)
   3. Interrupt 21H servis 02H (menuliskan karakter ASCII ke layar)
   4. Interrupt 21H servis 0AH (menerima input numeric string)
      Memanggil command **CSN,** dengan AX merupakan tempat penyimpanan numerik hasil konversi.
   5. Interrupt 21H servis 09H (mengeprint numeric string ke layar)
      Memanggil command **CNS,** dengan AX diisi dengan nilai yang ingin di-print.

2. MOV <register>,<immediate/register-2>
   Command MOV digunakan untuk **memindahkan** nilai **<immediate>** dalam bentuk hexadecimal atau nilai dalam **<register-2>** ke **<register-1>.**

3. ADD/SUB <register-1>,<register-2/immediate>
   Command ADD digunakan untuk **menjumlahkan** nilai <immediate> dalam bentuk hexadecimal atau nilai dalam **<register-2>** dengan **<register-1>** dan hasilnya disimpan dalam **<register-1>.** Secara matematis**, <register-1> = <register-1> + <immediate/register-2>.**
   Command SUB digunakan untuk **mengurangi** nilai **<register-1>** dengan **<immediate>** dalam bentuk hexadecimal atau nilai dalam **<register-2>** dan hasilnya disimpan dalam **<register-1>.** Secara matematis**, <register-1> = <register-1> - <immediate/register-2>.**

©Laboratorium Sistem Tersebar 2012

4. MUL/DIV <register-1_8 bit>
   Command MUL menerima masukan berupa nilai dalam register 8-bit, dan nilai akan dikalikan dengan nilai pada AL dan disimpan dalam dalam AX yang berukuran 16-bit.
   Contoh: MUL BH, maka rumus matematikanya dalam AX = AL x BH.
   Command DIV menerima masukan dari nilai pada register 8-bit, nilai tersebut akan dibagi dengan nilai pada AX dan hasilnya 8-bit akan disimpan dalam AL dan sisa pembagian akan disimpan dalam AH.
   Contoh: DIV BH, maka rumus matematikanya dalam AL = BH / AX, AH = BH mod AX.

5. AND/OR/XOR <register-1>,<register-2>
   Command AND/OR/XOR merupakan *bitwise* operation. Nilai secara bit pada **<register-1>** akan dilakukan logika AND/OR/XOR dengan nilai secara bit pada **<register-2>** dan disimpan hasilnya pada **<register-1>.**

6. NOT <register-1>
   Command NOT merupakan bitwise operation untuk operasi logika not. Nilai secara bit pada **<register-1>** akan dilakukan operasi logika not yang kemudian disimpan kembali pada **<register-1>**.

7. JMP <label>
   JMP adalah perintah untuk melakukan perpindahan ke alamat tertentu. Untuk mengimplementasikan alamat digunakan label dan baris dari program.

8. CMP <immediate/register-1>,<immediate/register-2>
   CMP adalah perintah untuk membandingkan 2 buah nilai. Kedua nilai tersebut dapat didapat dari register atau dari immediate value.

9. JL/JG/JGE/JLE <register-1>,<register-2>,<label>
   JL adalah perintah untuk melakukan perpindahan ke alamat tertentu jika nilai pada **<register-1> lebih kecil (JL)** atau **lebih besar (JG)** atau **lebih besar sama dengan (JGE)** atau **lebih kecil sama dengan (JLE)** dari nilai pada **<register-2>** pada perintah **CMP**. Untuk mengimplementasikan alamat digunakan label dan baris dari program.

10. LOOP <label>
    LOOP adalah perintah untuk melakukan perpindahan ke alamat tertentu jika nilai dari suatu register masih lebih besar dari 0. Dalam tugas ini, register yang digunakan untuk perintah LOOP adalah register CX. Setiap kali LOOP melakukan lompatan, maka nilai dari CX akan dikurang 1.

11. CALL <label>
    CALL berfungsi untuk melompat dan mengerjakan instruksi pada procedure program.

©Laboratorium Sistem Tersebar 2012

12. RET
    RET berfungsi untuk mengambil alamat pada stack untuk melakukan lompatan pada alamat tersebut. RET biasanya diletakkan pada akhir dari suatu procedure yang dipanggil dengan CALL.

13. CSN
    CSN bukan merupakan command utama yang bisa dituliskan secara langsung tetapi merupakan fungsi dipanggil dalam interrupt 21H service 0AH. Dalam hal ini, input berupa string desimal, contoh "8934", akan diubah dalam register sebagai number 8934.

14. CNS
    CNS bukan merupakan command utama yang bisa dituliskan secara langsung tetapi merupakan fungsi dipanggil dalam interrupt 21H service 09H. Dalam hal ini, input berupa nilai register, contoh 8934_{10}, akan diubah untuk di-print sebagai string "8934".

    Catatan: untuk command CSN dan CNS diimplementasi sebagai kode assembly pada file khusus (runtime.asm) yang akan di-load oleh interpreter sebelum membaca file program assembly. File ini seperti halnya library pada java.

File yang sudah dibaca kemudian diinterpret untuk kemudian dieksekusi langsung tanpa membuat executable filenya.

Bonus

1. Ada pemeriksaan kebenaran syntax command assembly. Jika ada kesalahan, berikan baris kesalahan terjadi dan pesan kesalahan.
   Contoh:

```
Proses:
MOV 02h,AH ; salah parameter
MOV DL,'A'
MOV CX,10h
Ulang:
INT 21h
INC DL
LOOP Ulang1 ; label tidak ada
INT 20h
```

2. eksekusi step-by-step (interaktif) dengan menambahkan argumen masukkan berupa '–i' dengan susunan **<nama_program><spasi>[-i]<spasi><nama_file>.** Contoh:
   `myasmc –i test.asm`
3. menampilkan isi (dump) register ke layar dengan menambahkan argumen masukkan berupa '–d' dengan susunan **<nama_program><spasi>[-d]<spasi><nama_file>.** Contoh:
   `myasmc –d test.asm`
4. kompilasi kode assembly menjadi biner (menggunakan memori virtual) dan menjalankan program dari memori dengan menambahkan argumen masukkan berupa '–c' dengan susunan **<nama_program><spasi>[-c]<spasi><nama_file>.** Contoh:
   `myasmc –d test.asm`
5. dump memory ke file teks setelah eksekusi dengan menambahkan argumen masukkan berupa '–m' dengan susunan **<nama_program><spasi>[-m]<spasi><nama_file>.** Contoh:

©Laboratorium Sistem Tersebar 2012

```
myasmc –m test.asm
```

Keterangan:
Apabila bonus kedua hingga bonus kelima lebih dari satu diimplementasikan, maka susunan argumen masukan pengguna berupa

<nama_program><spasi>[-i][spasi][-d][spasi][-c][spasi][-m]<spasi><nama_file>

Asistensi

Asistensi tugas besar 2 **wajib dilakukan minimal 1 kali**. Untuk asisten asistensi, silakan mengirimkan file zip dummy ke http://milestone.if.itb.ac.id dalam memilih jadwal pada jadwal demo. Asistensi paling lambat dilakukan hingga **20 April 2012** pukul **19:00**.

Deliverable

Tugas dikirim ke http://milestone.if.itb.ac.id dengan format .zip dengan nama TB2-KX-GYY.zip dengan X adalah nomor kelas dan YY adalah nomor grup. File yang perlu dikumpulkan adalah:

a. bin: berisi executable compiler
b. src: berisi source code compiler
c. doc: berisi readme cara menjalankan compiler dan laporan tugas dalam format .pdf

   Laporan berisi penjelasan program yang terdiri dari

   (1) Deskripsi Tugas Besar (Diperbolehkan untuk mengambil dari soal).
   (2) Implementasi, yang terdiri dari
       a. Penjelasan tentang pemodelan register dan memori dalam program
       b. Penjelasan tentang implementasi semua *command* yang diimplementasikan dalam program
       c. Struktur data dan fungsi/prosedur yang digunakan dalam implementasi
   (3) Fitur Program, termasuk fitur utama yang diimplementasikan dan fitur-fitur tambahan lainnya termasuk bonus.
   (4) Batasan Program, berisi batasan dari program.
   (5) Jawaban pertanyaan:
       a. Apa pendapat kelompok mengenai praktikum EL2010?

Deadline tugas besar ini adalah **29 April 2012**, pukul **20:10**.

Penilaian

Penilaian dilakukan terhadap kelompok berdasarkan hasil pengerjaan tugas dan laporan serta penilaian individu dari demo tugas. Pemotongan nilai akan diberikan jika terlambat mengumpulkan tugas dan / atau adanya kecurangan seperti mencontek atau sejenisnya.

☺ Selamat Mengerjakan ☺

©Laboratorium Sistem Tersebar 2012