KOM206 - ORGANISASI KOMPUTER
PENGENALAN BAHASA ASSEMBLY DAN TIPE REGISTER

TUJUAN PRAKTIKUM

1. Mahasiswa mampu menjelaskan fungsi CPU dan Register pada lingkungan x86
2. Mahasiswa dapat menjelaskan tipe-tipe dan fungsi register
3. Mahasiswa dapat membuat program sederhana menggunakan bahasa assembly

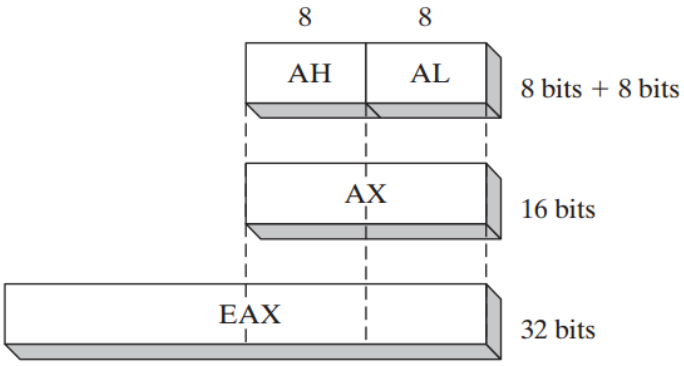
TEORI PENUNJANG

Memori komputer pada dasarnya merupakan array byte yang menggunakan perangkat lunak untuk instruksi dan data. Meskipun memori relatif cepat, namun ada kebutuhan untuk pemindahan sejumlah kecil data yang lebih cepat untuk memungkinkan CPU untuk menjalankan instruksi lebih cepat. Salah satu jenis memori yang lebih cepat adalah memori cache, yang mungkin 10 kali lebih cepat memori utama. Tipe kedua dari memori yang lebih cepat adalah register CPU. Cache berukuran beberapa megabyte, sementara CPU hanya memiliki beberapa register.

CPU x86-64 memiliki 16 general purpose 64 bit register dan 16 modern floating point modern register. 16 general purpose register berjumlah 64 bit yang tersimpan dalam CPU. Software dapat mengakses register dalam ukuran 64 bit , 32 bit, 16 bit dan 8 bit.

Pada register di 8088 CPU (16 bit) memiliki special purpose register selain general purpose register:

- ax - accumulator for numeric operations
- bx - base register (array access)
- ex - count register (string operations)
- dx - data register
- s i - source index
- di - destination index
- bp - base pointer (for function frames)
- sp - stack pointer



Gambar 1 General Purpose Registers

General purpose register secara umum digunakan untuk aritmetika dan pemnidahan data. Untuk mengakses low byte dari ax dapat menggunakan al, sementara untuk megakses high byte dari ax dapat menggunakan ah. Hal ini berlaku juga untuk register bx, cx, dan dx.

Ketika CPU 386 dibuat, register dilebarkan menjadi 32 bit dan diganti nama menjadi eax, ebx, ecx, edx, esi, edi, ebp, dan esp. Software dapat mengakses 16 bit terendah dengan menggunakan nama awalnya.

32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

Sementara, untuk beberapa register hanya dapat diakses dengan nama 32-bit dan 16-bit nya saja.

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

Beberapa general purpose register memiliki kegunaan khusus:

- EAX secara otomatis digunakan untuk instruksi perkalian dan pembagian.
- CPU secara otomatis menggunakan ECX sebagai loop counter
- Data alamat ESP pada stack biasanya digunakan sebagai extended stack pointer register.
- ESI dan EDI digunakan oleh instruksi transfer memori dengan kecepatan tinggi.
- EBP digunakan oleh high-level languages untuk mereferensi parameter fungsi dan lokal variabel dalam stack.

MATERI PRAKTIKUM

Dalam penggunaan bahasa assembly di X86, instruksi dituliskan dalam format seperti berikut:

LABEL: MNEMONIC OPERAND1, OPERAND2, OPERAND3

Label merupakan identifier untuk sebuah line kode ataupun blok kode, biasanya label diikuti dengan tanda colon (:). Mnemonic merupakan nama dari instruksi, misalnya ADD, SUB, MOV, dan lain-lain. Operand dalam instruksi dapat berjumlah dari nol hingga tiga operand, tergantung pada operasi / mnemonic yang digunakan. Setiap operand dipisahkan dengan tanda comma (,).

Mode pengalamatan memberikan berbagai cara untuk mengakses operand.

Mode pengalamatan

1. Register addressing

   Dalam mode pengalamatan ini, instruksi memilih satu atau lebih register yang merepresentasikan operand. Mode pengalamatan ini memiliki kecepatan eksekusi yang

tinggi karena seluruh operasi dilaksanakan didalam CPU.

Beberapa contoh register addressing:

```
INC BH        ; increment register BH
MOV AX, BX    ; memindahkan isi dari register BX ke register AX
```

Register sumber dan destinasi dapat dari berbagai general purpose register, baik 8-bit, 16-bit, 32-bit, atau 64-bit. Segment register CS, DS, SS, ES, FS, atau GS juga dapat digunakan dalam mode pengalamatan ini.

2. Immediate addressing

   Mode pengalamatan ini menggunakan nilai konstan dalam operasinya. Dalam instruksi dengan dua operand, operand pertama merupakan register atau lokasi memori sementara operand kedua merupakan nilai konstan.

```
MOV AX, 3H    ; memindahkan nilai 3 heksadesimal ke register AX
ADD AX, 5H    ; menambahkan nilai 5 heksadesimal ke register AX
```

3. Direct Memory Adressing

   Dalam direct memory addressing, salah satu operand merujuk ke sebuah lokasi memori dan operand lainnya mereferensikan ke register.

```
MOV AL, [1A33D4H] ; memindahkan nilai dalam memori 1A33D4H ke AL
```

4. Indirect Memory Addressing

   Indirect memory addressing biasanya digunakan ketika register terdiri dari alamat dari data, bukan nilai data yang akan diakses. Metode pengalamatan ini menggunakan general purpose register dalam format bracket, misalnya [BX], [BP], [EAX], [EBX], dan lain-lain.

```
MOV AX, [EBX] ; memindahkan nilai pada alamat EBX ke register AX
```

System Calls

System Calls adalah sebuah API untuk antarmuka antara user space dan kernel space. Berikut beberapa jenis system call:

```
mov    eax,1          ; system call number (sys_exit)
int    0x80           ; call kernel
```

%eax	Name	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	int	-	-	-	-
2	sys_fork	struct pt_regs	-	-	-	-

3	sys_read	unsigned int	char *	size_t	-	-
4	sys_write	unsigned int	const char *	size_t	-	-
5	sys_open	const char *	int	int	-	-
6	sys_close	unsigned int	-	-	-	-

Program 1. Hello, world!

```
segment  .text                ;code segment
    global _start                       ;must be declared for linker
_start:  ;tell linker entry point
        mov edx,len             ;message length
        mov ecx,msg             ;message to write
        mov ebx,1               ;file descriptor (stdout)
        mov eax,4               ;system call number (sys_write)
        int 0x80                ;call kernel
        mov eax,1               ;system call number (sys_exit)
        int 0x80                ;call kernel
segment .data                   ;data segment
msg     db 'Hello, world!',0xa   ;our dear string
len     equ     $ - msg         ;length of our dear string
```

Program 2. Menampilkan 9 bintang

```
section .text
    global  _start      ;must be declared for linker (gcc)
_start:  ;tell linker entry point
        mov     edx,len         ;message length
        mov     ecx,msg         ;message to write
        mov     ebx,1           ;file descriptor (stdout)
        mov     eax,4           ;system call number (sys_write)
        int     0x80            ;call kernel
        mov     edx,9           ;message length
        mov     ecx,s2          ;message to write
        mov     ebx,1           ;file descriptor (stdout)
        mov     eax,4           ;system call number (sys_write)
        int     0x80            ;call kernel
```

```
        mov     eax,1           ;system call number (sys_exit)
        int     0x80            ;call kernel
section .data
msg db 'Displaying 9 stars',0xa ;a message
len equ $ - msg                 ;length of message
```

```
s2 times 9 db '*'
```

Latihan

1. Buat program sederhana untuk:
   - menyimpan nilai 012h, 034h, dan 067h ke masing-masing offset 0150h, 0151h, dan 0152h
   - menjumlahkan ketiga nilai tersebut
   - menyimpan hasil penjumlahannya ke offset 0153h dan register AX.
2. Buat program sederhana untuk:
   - Menyimpan nilai 014h, 0FFh
   - Menjumlahkan nilai tersebut dan menyimpannya ke register
   - Hasil penjumlahan tersebut di kurangi dengan 03Ah
   - Hasil pengurangan disimpan ke register AX

DAFTAR PUSTAKA

[1] Seyfarth R. 2012. Introduction to 64 Bit Intel Assembly Language Programming for Linux. CreateSpace

[2] Irvine K. 2011. Assembly Language for X86 Processors. Pearson Education, Limited [Online]. Available: http://books.google.co.id/books?id=0k20RAAACAAJ

[3] Cavanagh J. 2013, X86 Assembly Language and C Fundamentals. CRC Press, 2013.

[4] http://www.tutorialspoint.com/assembly_programming