**LAPORANPRAKTIKUM SISTEM OPERASI MODUL 1**

****

Disusun oleh:

Nama : Leni Purwaningsih

NIM : L200190225

Kelas : F

**Tugas 1**

1. Apa yang dimaksud dengan kode “ASCI”, buatlah tabel kode ASCII lengkap cukup kode ASCII yang standar tidak perlu extended, tuliskan kode ASCII dalam format angka desimal, binary dan hexadesimal serta karakter dan simbol yang dikodekan.
2. Carilah daftar perintah bahasa asembly untuk mesin intel keluarga x86 lengkap(dari buku referensi atau internet ). daftar perintah ini dapat digunakan sebagai pedoman untuk memahami program “boot.asm” dan “kernel.asm”.
3. **Pengertian ASCII**

ASCII merupakan kepanjangan dari (American standard code for information interchange), dan pengertian dari ASCII sendiri adalah suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnnya 124 adalah untuk karakter “ “ dan biasa digunakan untuk komputer dan alat komunikasi lain untuk menunjukan teks.

**TABEL ASCII**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **biner** | **desimal** | **hexa** | **karakter** |  | **biner** | **des** | **hexa** | **karakter** |  | **biner** | **desimal** | **hexa** | **karakter** |
| 0 | 0 | 0 | NUL | 101011 | 43 | 2B | + | 1010110 | 86 | 56 | V |
| 1 | 1 | 1 | STX | 101100 | 44 | 2C | , | 1010111 | 87 | 57 | W |
| 10 | 2 | 2 | SOT | 101101 | 45 | 2D | - | 1011000 | 88 | 58 | X |
| 11 | 3 | 3 | ETX | 101110 | 46 | 2E | . | 1011001 | 89 | 59 | Y |
| 100 | 4 | 4 | EOT | 101111 | 47 | 2F | / | 1011010 | 90 | 5A | Z |
| 101 | 5 | 5 | ENQ | 110000 | 48 | 30 | 0 | 1011011 | 91 | 5B | [ |
| 110 | 6 | 6 | ACK | 110001 | 49 | 31 | 1 | 1011100 | 92 | 5C | \ |
| 111 | 7 | 7 | BEL | 110010 | 50 | 32 | 2 | 1011101 | 93 | 5D | ] |
| 1000 | 8 | 8 | BS | 110011 | 51 | 33 | 3 | 1011110 | 94 | 5E | ^ |
| 1001 | 9 | 9 | HT | 110100 | 52 | 34 | 4 | 1011111 | 95 | 5F | \_ |
| 1010 | 10 | A | LF | 110101 | 53 | 35 | 5 | 1100000 | 96 | 60 | ` |
| 1011 | 11 | B | VT | 110110 | 54 | 36 | 6 | 1100001 | 97 | 61 | a |
| 1100 | 12 | C | FF | 110111 | 55 | 37 | 7 | 1100010 | 98 | 62 | b |
| 1101 | 13 | D | CR | 111000 | 56 | 38 | 8 | 1100011 | 99 | 63 | c |
| 1110 | 14 | E | SO | 111001 | 57 | 39 | 9 | 1100100 | 100 | 64 | d |
| 1111 | 15 | F | SI | 111010 | 58 | 3A | : | 1100101 | 101 | 65 | e |
| 10000 | 16 | 10 | DLE | 111011 | 59 | 3B | ; | 1100110 | 102 | 66 | f |
| 10001 | 17 | 11 | DC1 | 111100 | 60 | 3C | < | 1100111 | 103 | 67 | g |
| 10010 | 18 | 12 | DC2 | 111101 | 61 | 3D | = | 1101000 | 104 | 68 | h |
| 10011 | 19 | 13 | DC3 | 111110 | 62 | 3E | > | 1101001 | 105 | 69 | i |
| 10100 | 20 | 14 | DC4 | 111111 | 63 | 3F | ? | 1101010 | 106 | 6A | j |
| 10101 | 21 | 15 | NAK | 1000000 | 64 | 40 | @ | 1101011 | 107 | 6B | k |
| 10110 | 22 | 16 | SYN | 1000001 | 65 | 41 | A | 1101100 | 108 | 6C | l |
| 10111 | 23 | 17 | ETB | 1000010 | 66 | 42 | B | 1101101 | 109 | 6D | m |
| 11000 | 24 | 18 | CAN | 1000011 | 67 | 43 | C | 1101110 | 110 | 6E | n |
| 11001 | 25 | 19 | EM | 1000100 | 68 | 44 | D | 1101111 | 111 | 6F | o |
| 11010 | 26 | 1A | SUB | 1000101 | 69 | 45 | E | 1110000 | 112 | 70 | p |
| 11011 | 27 | 1B | ESC | 1000110 | 70 | 46 | F | 1110001 | 113 | 71 | q |
| 11100 | 28 | 1C | FS | 1000111 | 71 | 47 | G | 1110010 | 114 | 72 | r |
| 11101 | 29 | 1D | GS | 1001000 | 72 | 48 | H | 1110011 | 115 | 73 | s |
| 11110 | 30 | 1E | RS | 1001001 | 73 | 49 | I | 1110100 | 116 | 74 | t |
| 11111 | 31 | 1F | US | 1001010 | 74 | 4A | J | 1110101 | 117 | 75 | u |
| 100000 | 32 | 20 | SP | 1001011 | 75 | 4B | K | 1110110 | 118 | 76 | v |
| 100001 | 33 | 21 | ! | 1001100 | 76 | 4C | L | 1110111 | 119 | 77 | w |
| 100010 | 34 | 22 | " | 1001101 | 77 | 4D | M | 1111000 | 120 | 78 | x |
| 100011 | 35 | 23 | # | 1001110 | 78 | 4E | N | 1111001 | 121 | 79 | y |
| 100100 | 36 | 24 | $ | 1001111 | 79 | 4F | O | 1111010 | 122 | 7A | z |
| 100101 | 37 | 25 | % | 1010000 | 80 | 50 | P | 1111011 | 123 | 7B | { |
| 100110 | 38 | 26 | & | 1010001 | 81 | 51 | Q | 1111100 | 124 | 7C | | |
| 100111 | 39 | 27 | ' | 1010010 | 82 | 52 | R | 1111101 | 125 | 7D | } |
| 101000 | 40 | 28 | ( | 1010011 | 83 | 53 | S | 1111110 | 126 | 7E | ~ |
| 101001 | 41 | 29 | ) | 1010100 | 84 | 54 | T | 1111111 | 127 | 7F | DEL |
| 101010 | 42 | 2A | \* | 1010101 | 85 | 55 | U |  | | | | |

1. **Daftar Perintah Bahasa Assembly x86**

Bahasa Assembly adalah bahasa level rendah dari bahasa pemrograman. CPU 8086

mempunyai 8 general purpose register, diantarannya:

* **AX** :the accumulator register (AH dan AL)
* **BX** :the base address register (BH dan BL)
* **CX** :the count register (dibagi menjadi CH dan CL)
* **DX** :the data register (dibagi menjadi DH dan DL)
* **SI** :source index register
* **DI** :destination index register
* **BP** :base pointer
* **SP** :stack pointer

Mikroprosesor 8086 memiliki dua mode proses CPU yaitu mode minimum dan mode maksimum. Pada mikroprosesor x86 (prosesor diatas 80386) memiliki dua mode proses juga yaitu normal mode dan protected mode. Prosesor ini adalah prosesor 32 bit.

Berikut penjelasan Bahasa Assembly-nya:

1. **Komentar**

Komentar diawali dengan tanda titik koma (;) ; ini adalah komentar

1. **Label**

Label diakhiri dengan tanda titik dua (contoh: main; loop; proses; keluar.

1. **Assembler directives**

- *Directives*, adalah perintah yang ditujukan kepada assembler ketika sedang

menerjemahkan program kita ke bahasa mesin. Directive dimulai dengan tanda titik

- *Model* : memberitahu assembler berapa memori yang akan dipakai oleh program.

Ada model tiny, model small, model compact, model medium, model large, dan

model huge.

- *Data* : memberitahu assembler bahwa bagian di bawah ini adalah data program.

- *Code* : memberitahu assembler bahwa bagian di bawah ini adalah instruksi program.

- *Stack* : memberitahu assembler bahwa program kita memiliki stack.

1. **Definisi Data**

- *DB* : define bytes. Membentuk data byte demi byte. Data bisa data numerik maupun

teks. Contoh: teks 1 db “Hello world $”.

- *DW* : define words. Membentuk data word demi word (1 word = 2 byte). Syntax:

{label} DW {data}. Contoh: kucing dw ?, ?, ?; mendefinisikan tiga slot 16-bit yang

isinya don’t care (disimbolkan dengan tanda tanya).

- *DD* : define double words. Membentuk data doubleword demi doubleword (4 byte).

Syntax: {label} DD {data}

- *EQU* : equals. Membentuk konstanta. Syntax: {label} EQU {data}. Contoh: sepuluh

EQU 10 untuk assembly yang melibatkan bilangan pecahan (floating point).

1. **Perpindahan data**

- *MOV* : move. Memindahkan suatu nilai dari register ke memori, memori ke register,

atau register ke register. Syntax: MOV {tujuan}, {sumber}. Contoh mov AX,

4C00H; mengisi register AX dengan 4C00(hex).

- *LEA* : load effective address. Mengisi suatu register dengan alamat offset sebuah

data. Syntax: LEA {register}, {sumber}. Contoh: lea DX, teks 1.

- *XCHG* : exchange. Menukar dua buah register langsung. Syntax: XCHG {register

1}, {register 2} Kedua register harus punya ukuran yang sama. Bila sama-sama 8

bit (misalnya AH dengan BL) atau sama-sama 16 bit (misalnya CX dan DX), maka

pertukaran bisa dilakukan.

1. **Operasi Logika**

- *AND* : melakukan bitwise and. Sintaks: AND {register}, {angka} AND {register 1},

{register 2} hasil disimpan di register 1. Contoh: mov AL, 00001011b mov AH,

11001000b and AL, AH ; sekarang AL berisi 00001000(bin), sedangkan AH tidak

berubah.

- *OR* : melakukan bitwise or. Sintaks: OR {register}, {angka} OR {register 1},

{register 2} hasil disimpan di register 1.

- *NOT* : melakukan bitwise not (one’s complement). Sintaks: NOT {register} hasil

disimpan di register itu sendiri.

1. **Operasi matematika**

- *ADD* : add. Menjumlahkan dua buah register. Sintaks: ADD {tujuan}, {sumber}

operasi yang terjadi: tujuan = tujuan + sumber. carry (bila ada) disimpan di CF.

- *INC* : increment. Menjumlah isi sebuah register dengan 1. Bedanya dengan ADD,

perintah INC hanya memakan 1 byte memori sedangkan ADD pakai 3 byte. Sintaks:

INC {register}

- *SUB* : substract. Mengurangkan dua buah register. Sintaks: SUB {tujuan}.

{sumber} operasi yang terjadi: tujuan = tujuan – sumber. borrow (bila terjadi)

menyebabkan CF bernilai 1.

- *DEC* : decrement. Mengurang isi sebuah register dengan 1. Jika SUB memakai 3

byte memori, DEC hanya memakai 1 byte. Sintaks: DEC {register}.

- *MUL* : multiply. Mengalikan register dengan AX atau AH. Sintaks: MUL {sumber}

1. **Pengulangan**

- *LOOP* : loop. Mengulang sebuah proses. Sintaks: LOOP {label tujuan}

- *LOOPE* : loop while equal. Melakukan pengulangan selama CX ≠ 0 dan ZF = 1. CX

tetap dikurangi 1 sebelum diperiksa. Sintaks: LOOP {label tujuan}.

1. **Perbandingan**

- *CMP* : compare. Membandingkan dua buah operand. Hasilnya mempengaruhi

sejumlah flag register. Sintaks: CMP {operand 1}, {operand 2}.

1. **Lompat-lompat**

- *JMP*: jump. Lompat tanpa syarat. Sintaks: JMP {label tujuan} Lompat bersyarat sintaksnya sama dengan JMP, yaitu perintah jump diikuti label tujuan.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Perintah** | **Arti** | **syarat** | **Kasus** | **Ket. (“op”= operand)** | **Mengikuti CMP?** |
|
| **JA** | Jump if above | CF = 0 **^** ZF  = 0 | unsigned | Lompat bila op 1 > op 2 | Ya |
| **JNBE** | Jump if not below or equal |
| **JB** | Jump if below | CF =1 **^** ZF  = 0 | unsigned | Lompat bila op 1 < op 2 | Ya |
| **JNAE** | Jump if not above or equal |
| **JAE** | Jump if above or equal | CF = 0 v ZF  = 1 | unsigned | Lompat bila op 1 > op 2 | Ya |
| **JNB** | Jump if not below |
| **JBE** | Jump if below or equal | CF = 1 v ZF  = 1 | unsigned | Lompat bila op 1 < op 2 | Ya |
| **JNA** | Jump if not above |
| **JG** | Jump if greater | OF = 0 ^ ZF  = 0 | signed | Lompat bila op 1 > op 2 | Ya |
| **JNLE** | Jump if not less or equal |
| **JGE** | Jump if greater or equal | OF = 0 v ZF  = 1 | signed | Lompat bila op 1 > op 2 | Ya |
| **JNL** | Jump if not less than |
| **JL** | Jump if less than | OF = 1 ^ ZF  = 0 | signed | Lompat bila op 1 < 2 op 2 | Ya |
| **JNGE** | Jump if not greater or equal |
| **JLE** | Jump if less or equal | OF =1 v ZF  =1 | signed | Lompat bila op 1 < op 2 | Ya |
| **JNG** | Jump if not greater |
| **JE** | Jump if equal | ZF = 1 | keduanya | Lompat bila op = op 2 | Ya |
| **JZ** | Jump if zero | ZF = 1 | Keduanya | Lompat bila op 1 = op 2 | Ya |
| **JNE** | Jump if not equal | ZF = 0 | Keduanya | Lompat bila op 1 ≠ op 2 | Ya |
| **JNZ** | Jump if not zero | ZF = 0 | keduanya | Lompat bila op ≠ op 2 | Ya |
| **JC** | Jump if carry | CF = 1 | N/A | Lompat bila carry flag = 1 | Tidak |
| **JNC** | Jump if not carry | CF = 0 | N/A | Lompat bila carry flag = 0 | Tidak |
| **JP** | Jump on parity | PF = 1 | N/A | Lompat bila parity flag = 1 | Tidak selalu |
| **JPE** | Jump on parity even | Lompat bila bilangan genap | Tidak selalu |
| **JNP** | Jump on not parity | PF = 0 | N/A | Lompat bila parity flag = 0 | Tidak selalu |
| **JNO** | Jump if not overflow | OF = 0 | N/A | Lompat bila overflow  flag = 0 | Tidak |
| **JS** | Jump if sign | SF = 1 | N/A | Lompat bila bilangan negatif | Tidak |
| **JCXZ** | Jump if CX is zero | CX = 0000 | N/A | Lompat bila CX berisi nol | Tidak |

1. **Operasi stack**

- Push : Menambahkan sesuatu ke stack. Sesuatu ini harus register berukuran 16

bit(pada 386+ harus 32 bit) tidak boleh menggunakan angka , tidak boleh

menggunakan alamat memori, jika terjadi maka anda tidak dapat meng-push

register 8-bit misalnnya : AH, AL, BH, dan BL.

- Pop :Mengambil dari stack, maka akan disimpan di register dengan tujuan

16 - bit. Jika demikian maka tidak dapat mem-pop menuju AH, dan AL.

- Pushf : merupakan pushf flags. mem-push isi dari register flag ke dalam

stack.bisa membackup data di register flag sebelum operasi matematika.

- popf : pop flags. Lawan dari pushf sintaks.

- popa : pop all general-purpose register. Adalah ringkasan dari sejumlah perintah

dengan urutan pop DI pop SI pop BP pop SP pop BX pop DX pop CX pop CX

pop AX.

- Pusha : push all general-purpose registers. Lawan dari POPA, dimana pusha adalah

singkatan dari sejumlah perintah dengan urutan yang sudah ditetapkan.