Hanif Firdaus Illahi

L200180021 (A)

Tugas praktikum SO 1

**1. PENGERTIAN KODE ASCII**

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|"



**Definisi data**

**DB** : define bytes. Membentuk data byte demi byte. Data bisa data numerik maupun teks. catatan: untuk membentuk data string, pada akhir string harus diakhiri tanda dolar ($). sintaks: {label} DB {data} contoh: teks1 db "Hello world $" **DW** : define words. Membentuk data word demi word (1 word = 2 byte). sintaks: {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). sintaks: {label} DD {data}

**EQU** : equals. Membentuk konstanta. sintaks: {label} EQU {data} contoh: sepuluh EQU 10 Ada assembly yang melibatkan bilangan pecahan (floating point), bilangan bulat (integer), DF (define far words), DQ (define quad words), dan DT (define ten bytes).

**Perpindahan data**

**MOV** : move. Memindahkan suatu nilai dari register ke memori, memori ke register, atau register ke register. Sintaks: MOV {tujuan}, {sumber}

contoh:

*mov AX, 4C00h ;mengisi register AX dengan 4C00(hex).*

*mov BX, AX ;menyalin isi AX ke BX. mov CL, [BX] ;mengisi register CL dengan data di memori yang alamatnya ditunjuk BX.*

*mov CL, [BX] + 2 ;mengisi CL dengan data di memori yang alamatnya ditunjuk BX lalu geser maju 2 byte.*

*mov [BX], AX ;menyimpan nilai AX pada tempat di memori yang ditunjuk BX. mov [BX] – 1, 00101110b ;menyimpan 00101110(bin) pada alamat yang ditunjuk BX lalu geser mundur 1 byte.*

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

sintaks: LEA {register}, {sumber} contoh: lea DX, teks1 **XCHG** : exchange. Menukar dua buah register langsung. sintaks: 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. Sebenarnya masih banyak perintah perpindahan data, misalnya IN, OUT, LODS, LODSB, LODSW, MOVS, MOVSB, MOVSW, LDS, LES, LAHF, SAHF, dan XLAT.

**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.

**XOR** : melakukan bitwise eksklusif or. sintaks: XOR {register}, {angka} XOR {register 1}, {register 2} hasil disimpan di register 1. Tips: sebuah register yang di-XOR-kan dengan dirinya sendiri akan menjadi berisi nol.

**SHL** : shift left. Menggeser bit ke kiri. Bit paling kanan diisi nol. sintaks: SHL {register}, {banyaknya}

**SHR** : shift right. Menggeser bit ke kanan. Bit paling kiri diisi nol. sintaks: SHR {register}, {banyaknya}

**ROL** : rotate left. Memutar bit ke kiri. Bit paling kiri jadi paling kanan kali ini. sintaks: ROL {register}, {banyaknya} Bila banyaknya rotasi tidak disebutkan, maka nilai yang ada di CL akan digunakan sebagai banyaknya rotasi.

**ROR** : rotate right. Memutar bit ke kanan. Bit paling kanan jadi paling kiri. sintaks: ROR {register}, {banyaknya} Bila banyaknya rotasi tidak disebutkan, maka nilai yang ada di CL akan digunakan sebagai banyaknya rotasi. Ada lagi : RCL dan RCR.

**Operasi matematika**

**ADD** : add. Menjumlahkan dua buah register. Sintaks: ADD {tujuan}, {sumber} operasi yang terjadi: tujuan = tujuan + sumber. Carry (bila ada) disimpan di CF.

**ADC** : add with carry. Menjumlahkan dua register dan carry flag (CF). Sintaks: ADC {tujuan}, {sumber} operasi yang terjadi: tujuan = tujuan + sumber + CF. carry (bila ada lagi) disimpan lagi 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.

**SBB** : substract with borrow. Mengurangkan dua register dan carry flag (CF). Sintaks: SBB {tujuan}, {sumber} operasi yang terjadi: tujuan = tujuan – sumber – CF. Borrow (bila terjadi lagi) menyebabkan CF dan SF (sign flag) 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} Bila register sumber adalah 8 bit, maka isi register itu dikali dengan isi AL, kemudian disimpan di AX. Bila register sumber adalah 16 bit, maka isi register itu dikali dengan isi AX, kemudian hasilnya disimpan di DX:AX. Maksudnya, DX berisi high order byte-nya, AX berisi low order byte-nya.

**IMUL** : signed multiply. Sama dengan MUL, hanya saja IMUL menganggap bit-bit yang ada di register sumber sudah dalam bentuk *two’s complement*. Sintaks: IMUL {sumber}

**DIV** : divide. Membagi AX atau DX:AX dengan sebuah register. Sintaks: DIV {sumber} Bila register sumber adalah 8 bit (misalnya: BL), maka operasi yang terjadi: -AX dibagi BL, -hasil bagi disimpan di AL, -sisa bagi disimpan di AH. Bila register sumber adalah 16 bit (misalnya: CX), maka operasi yang terjadi: -DX:AX dibagi CX, -hasil bagi disimpan di AX, -sisa bagi disimpan di DX.

**IDIV** : signed divide. Sama dengan DIV, hanya saja IDIV menganggap bit-bit yang ada di register sumber sudah dalam bentuk *two’s complement*. Sintaks: IDIV {sumber}

**NEG** : negate. Membuat isi register menjadi negatif (*two’s complement*). Bila mau *one’s complement*, gunakan perintah NOT. sintaks: NEG {register} hasil disimpan di register itu sendiri.