



















TUGAS:

1. Apa yang dimaksud dengan kode ASCII, buatlah tabel kode ASCII lengkap cukup kode ASCII yang standar tidak perlu extended, tuliskan kode ASCII dalam format angka desimal, binnary, dan heksadesimal serta karakter dan simbol yang dikodekan.

**Jawab : kode Standar Amerika untuk Pertukaran Informasi** atau *American Standard Code for Information Interchange* (**ASCII**) merupakan suatu standar internasional dalam kode [huruf](https://id.wikipedia.org/wiki/Huruf) dan [simbol](https://id.wikipedia.org/wiki/Simbol) seperti [Hex](https://id.wikipedia.org/wiki/Hex) dan [Unicode](https://id.wikipedia.org/wiki/Unicode) tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh [komputer](https://id.wikipedia.org/wiki/Komputer) dan alat komunikasi lain untuk menunjukkan teks

|  |  |  |  |
| --- | --- | --- | --- |
| **Karakter** | **Nilai Unicode (heksadesimal)** | **Nilai ANSI ASCII (desimal)** | **Keterangan** |
| NUL | 0000 | [0](https://id.wikipedia.org/wiki/0_(angka)) | Null (tidak tampak) |
| SOH | 0001 | [1](https://id.wikipedia.org/wiki/1_(angka)) | Start of heading (tidak tampak) |
| STX | 0002 | [2](https://id.wikipedia.org/wiki/2_(angka)) | Start of text (tidak tampak) |
| ETX | 0003 | [3](https://id.wikipedia.org/wiki/3_(angka)) | End of text (tidak tampak) |
| EOT | 0004 | [4](https://id.wikipedia.org/wiki/4_(angka)) | End of transmission (tidak tampak) |
| ENQ | 0005 | [5](https://id.wikipedia.org/wiki/5_(angka)) | Enquiry (tidak tampak) |
| ACK | 0006 | [6](https://id.wikipedia.org/wiki/6_(angka)) | Acknowledge (tidak tampak) |
| BEL | 0007 | [7](https://id.wikipedia.org/wiki/7_(angka)) | Bell (tidak tampak) |
| BS | 0008 | [8](https://id.wikipedia.org/wiki/8_(angka)) | Menghapus satu karakter di belakang kursor (Backspace) |
| HT | 0009 | [9](https://id.wikipedia.org/wiki/9_(angka)) | Horizontal tabulation |
| LF | 000A | [10](https://id.wikipedia.org/wiki/10_(angka)) | Pergantian baris (Line feed) |
| VT | 000B | [11](https://id.wikipedia.org/wiki/11_(angka)) | Tabulasi vertikal |
| FF | 000C | [12](https://id.wikipedia.org/wiki/12_(angka)) | Pergantian baris (Form feed) |
| CR | 000D | [13](https://id.wikipedia.org/wiki/13_(angka)) | Pergantian baris (carriage return) |
| SO | 000E | [14](https://id.wikipedia.org/wiki/14_(angka)) | Shift out (tidak tampak) |
| SI | 000F | [15](https://id.wikipedia.org/wiki/15_(angka)) | Shift in (tidak tampak) |
| DLE | 0010 | [16](https://id.wikipedia.org/wiki/16_(angka)) | Data link escape (tidak tampak) |
| DC1 | 0011 | [17](https://id.wikipedia.org/wiki/17_(angka)) | Device control 1 (tidak tampak) |
| DC2 | 0012 | [18](https://id.wikipedia.org/wiki/18_(angka)) | Device control 2 (tidak tampak) |
| DC3 | 0013 | [19](https://id.wikipedia.org/wiki/19_(angka)) | Device control 3 (tidak tampak) |
| DC4 | 0014 | [20](https://id.wikipedia.org/wiki/20_(angka)) | Device control 4 (tidak tampak) |
| NAK | 0015 | [21](https://id.wikipedia.org/wiki/21_(angka)) | Negative acknowledge (tidak tampak) |
| SYN | 0016 | [22](https://id.wikipedia.org/wiki/22_(angka)) | Synchronous idle (tidak tampak) |
| ETB | 0017 | [23](https://id.wikipedia.org/wiki/23_(angka)) | End of transmission block (tidak tampak) |
| CAN | 0018 | [24](https://id.wikipedia.org/wiki/24_(angka)) | Cancel (tidak tampak) |
| EM | 0019 | [25](https://id.wikipedia.org/wiki/25_(angka)) | End of medium (tidak tampak) |
| SUB | 001A | [26](https://id.wikipedia.org/wiki/26_(angka)) | Substitute (tidak tampak) |
| ESC | 001B | [27](https://id.wikipedia.org/wiki/27_(angka)) | Escape (tidak tampak) |
| FS | 001C | [28](https://id.wikipedia.org/wiki/28_(angka)) | File separator |
| GS | 001D | [29](https://id.wikipedia.org/wiki/29_(angka)) | Group separator |
| RS | 001E | [30](https://id.wikipedia.org/wiki/30_(angka)) | Record separator |
| US | 001F | [31](https://id.wikipedia.org/wiki/31_(angka)) | Unit separator |
| SP | 0020 | [32](https://id.wikipedia.org/wiki/32_(angka)) | Spasi |
| ! | 0021 | [33](https://id.wikipedia.org/wiki/33_(angka)) | Tanda seru (exclamation) |
| " | 0022 | [34](https://id.wikipedia.org/wiki/34_(angka)) | Tanda kutip dua |
| # | 0023 | [35](https://id.wikipedia.org/wiki/35_(angka)) | Tanda pagar (kres) |
| $ | 0024 | [36](https://id.wikipedia.org/wiki/36_(angka)) | Tanda mata uang dolar |
| % | 0025 | [37](https://id.wikipedia.org/wiki/37_(angka)) | Tanda persen |
| & | 0026 | [38](https://id.wikipedia.org/wiki/38_(angka)) | Karakter ampersand (&) |
| ‘ | 0027 | [39](https://id.wikipedia.org/wiki/39_(angka)) | Karakter Apostrof |
| ( | 0028 | [40](https://id.wikipedia.org/wiki/40_(angka)) | Tanda kurung buka |
| ) | 0029 | [41](https://id.wikipedia.org/wiki/41_(angka)) | Tanda kurung tutup |
| \* | 002A | [42](https://id.wikipedia.org/wiki/42_(angka)) | Karakter asterisk (bintang) |
| + | 002B | [43](https://id.wikipedia.org/wiki/43_(angka)) | Tanda tambah (plus) |
| , | 002C | [44](https://id.wikipedia.org/wiki/44_(angka)) | Karakter koma |
| - | 002D | [45](https://id.wikipedia.org/wiki/45_(angka)) | Karakter hyphen (strip) |
| . | 002E | [46](https://id.wikipedia.org/wiki/46_(angka)) | Tanda titik |
| / | 002F | [47](https://id.wikipedia.org/wiki/47_(angka)) | Garis miring (*slash*) |
| 0 | 0030 | [48](https://id.wikipedia.org/wiki/48_(angka)) | Angka nol |
| 1 | 0031 | [49](https://id.wikipedia.org/wiki/49_(angka)) | Angka satu |
| 2 | 0032 | [50](https://id.wikipedia.org/wiki/50_(angka)) | Angka dua |
| 3 | 0033 | [51](https://id.wikipedia.org/wiki/51_(angka)) | Angka tiga |
| 4 | 0034 | [52](https://id.wikipedia.org/wiki/52_(angka)) | Angka empat |
| 5 | 0035 | [53](https://id.wikipedia.org/wiki/53_(angka)) | Angka lima |
| 6 | 0036 | [54](https://id.wikipedia.org/wiki/54_(angka)) | Angka enam |
| 7 | 0037 | [55](https://id.wikipedia.org/wiki/55_(angka)) | Angka tujuh |
| 8 | 0038 | [56](https://id.wikipedia.org/wiki/56_(angka)) | Angka delapan |
| 9 | 0039 | [57](https://id.wikipedia.org/wiki/57_(angka)) | Angka sembilan |
| : | 003A | [58](https://id.wikipedia.org/wiki/58_(angka)) | Tanda titik dua |
| ; | 003B | [59](https://id.wikipedia.org/wiki/59_(angka)) | Tanda titik koma |
| < | 003C | [60](https://id.wikipedia.org/wiki/60_(angka)) | Tanda lebih kecil |
| = | 003D | [61](https://id.wikipedia.org/wiki/61_(angka)) | Tanda sama dengan |
| > | 003E | [62](https://id.wikipedia.org/wiki/62_(angka)) | Tanda lebih besar |
| ? | 003F | [63](https://id.wikipedia.org/wiki/63_(angka)) | Tanda tanya |
| @ | 0040 | [64](https://id.wikipedia.org/wiki/64_(angka)) | A keong (@) |
| A | 0041 | [65](https://id.wikipedia.org/wiki/65_(angka)) | Huruf latin A kapital |
| B | 0042 | [66](https://id.wikipedia.org/wiki/66_(angka)) | Huruf latin B kapital |
| C | 0043 | [67](https://id.wikipedia.org/wiki/67_(angka)) | Huruf latin C kapital |
| D | 0044 | [68](https://id.wikipedia.org/wiki/68_(angka)) | Huruf latin D kapital |
| E | 0045 | [69](https://id.wikipedia.org/wiki/69_(angka)) | Huruf latin E kapital |
| F | 0046 | [70](https://id.wikipedia.org/wiki/70_(angka)) | Huruf latin F kapital |
| G | 0047 | 71 | Huruf latin G kapital |
| H | 0048 | [72](https://id.wikipedia.org/wiki/72_(angka)) | Huruf latin H kapital |
| I | 0049 | 73 | Huruf latin I kapital |
| J | 004A | 74 | Huruf latin J kapital |
| K | 004B | [75](https://id.wikipedia.org/wiki/75_(angka)) | Huruf latin K kapital |
| L | 004C | 76 | Huruf latin L kapital |
| M | 004D | 77 | Huruf latin M kapital |
| N | 004E | [78](https://id.wikipedia.org/wiki/78_(angka)) | Huruf latin N kapital |
| O | 004F | [79](https://id.wikipedia.org/wiki/79_(angka)) | Huruf latin O kapital |
| P | 0050 | [80](https://id.wikipedia.org/wiki/80_(angka)) | Huruf latin P kapital |
| Q | 0051 | [81](https://id.wikipedia.org/wiki/81_(angka)) | Huruf latin Q kapital |
| R | 0052 | 82 | Huruf latin R kapital |
| S | 0053 | 83 | Huruf latin S kapital |
| T | 0054 | 84 | Huruf latin T kapital |
| U | 0055 | [85](https://id.wikipedia.org/wiki/85_(angka)) | Huruf latin U kapital |
| V | 0056 | 86 | Huruf latin V kapital |
| W | 0057 | 87 | Huruf latin W kapital |
| X | 0058 | [88](https://id.wikipedia.org/wiki/88_(angka)) | Huruf latin X kapital |
| Y | 0059 | 89 | Huruf latin Y kapital |
| Z | 005A | [90](https://id.wikipedia.org/wiki/90_(angka)) | Huruf latin Z kapital |
| [ | 005B | [91](https://id.wikipedia.org/wiki/91_(angka)) | Kurung siku kiri |
| \ | 005C | [92](https://id.wikipedia.org/wiki/92_(angka)) | Garis miring terbalik (*backslash*) |
| ] | 005D | 93 | Kurung sikur kanan |
| ^ | 005E | 94 | Tanda pangkat |
| \_ | 005F | [95](https://id.wikipedia.org/wiki/95_(angka)) | Garis bawah (underscore) |
| ` | 0060 | 96 | Tanda petik satu |
| a | 0061 | 97 | Huruf latin a kecil |
| b | 0062 | [98](https://id.wikipedia.org/wiki/98_(angka)) | Huruf latin b kecil |
| c | 0063 | [99](https://id.wikipedia.org/wiki/99_(angka)) | Huruf latin c kecil |
| d | 0064 | [100](https://id.wikipedia.org/wiki/100_(angka)) | Huruf latin d kecil |
| e | 0065 | 101 | Huruf latin e kecil |
| f | 0066 | 102 | Huruf latin f kecil |
| g | 0067 | 103 | Huruf latin g kecil |
| h | 0068 | 104 | Huruf latin h kecil |
| i | 0069 | 105 | Huruf latin i kecil |
| j | 006A | 106 | Huruf latin j kecil |
| k | 006B | 107 | Huruf latin k kecil |
| l | 006C | 108 | Huruf latin l kecil |
| m | 006D | 109 | Huruf latin m kecil |
| n | 006E | 110 | Huruf latin n kecil |
| o | 006F | 111 | Huruf latin o kecil |
| p | 0070 | 112 | Huruf latin p kecil |
| q | 0071 | 113 | Huruf latin q kecil |
| r | 0072 | 114 | Huruf latin r kecil |
| s | 0073 | 115 | Huruf latin s kecil |
| t | 0074 | 116 | Huruf latin t kecil |
| u | 0075 | 117 | Huruf latin u kecil |
| v | 0076 | 118 | Huruf latin v kecil |
| w | 0077 | 119 | Huruf latin w kecil |
| x | 0078 | 120 | Huruf latin x kecil |
| y | 0079 | 121 | Huruf latin y kecil |
| z | 007A | 122 | Huruf latin z kecil |
| { | 007B | 123 | Kurung kurawal buka |
| ¦ | 007C | 124 | Garis vertikal (pipa) |
| } | 007D | 125 | Kurung kurawal tutup |
| ~ | 007E | 126 | Karakter gelombang (tilde) |
| DEL | 007F | 127 | Delete |
|  | 0080 | 128 | Dicadangkan |
|  | 0081 | 129 | Dicadangkan |
|  | 0082 | 130 | Dicadangkan |
|  | 0083 | 131 | Dicadangkan |
| IND | 0084 | 132 | Index |
| NEL | 0085 | 133 | Next line |
| SSA | 0086 | 134 | Start of selected area |
| ESA | 0087 | 135 | End of selected area |
|  | 0088 | 136 | Character tabulation set |
|  | 0089 | 137 | Character tabulation with justification |
|  | 008A | 138 | Line tabulation set |
| PLD | 008B | 139 | Partial line down |
| PLU | 008C | 140 | Partial line up |
|  | 008D | 141 | Reverse line feed |
| SS2 | 008E | 142 | Single shift two |
| SS3 | 008F | 143 | Single shift three |
| DCS | 0090 | 144 | Device control string |
| PU1 | 0091 | 145 | Private use one |
| PU2 | 0092 | 146 | Private use two |
| STS | 0093 | 147 | Set transmit state |
| CCH | 0094 | 148 | Cancel character |
| MW | 0095 | 149 | Message waiting |
|  | 0096 | 150 | Start of guarded area |
|  | 0097 | 151 | End of guarded area |
|  | 0098 | 152 | Start of string |
|  | 0099 | 153 | Dicadangkan |
|  | 009A | 154 | Single character introducer |
| CSI | 009B | 155 | Control sequence introducer |
| ST | 009C | 156 | String terminator |
| OSC | 009D | 157 | Operating system command |
| PM | 009E | 158 | Privacy message |
| APC | 009F | 158 | Application program command |
|  | 00A0 | 160 | Spasi yang bukan pemisah kata |
| ¡ | 00A1 | 161 | Tanda seru terbalik |
| ¢ | 00A2 | 162 | Tanda sen (Cent) |
| £ | 00A3 | 163 | Tanda Poundsterling |
| ¤ | 00A4 | 164 | Tanda mata uang (*Currency*) |
| ¥ | 00A5 | 165 | Tanda Yen |
| ¦ | 00A6 | 166 | Garis tegak putus-putus (*broken bar*) |
| § | 00A7 | 167 | Section sign |
| ¨ | 00A8 | 168 | Diaeresis |
| © | 00A9 | 169 | Tanda hak cipta (Copyright) |
| ª | 00AA | 170 | Feminine ordinal indicator |
| « | 00AB | 171 | Left-pointing double angle quotation mark |
| ¬ | 00AC | 172 | Not sign |
|  | 00AD | 173 | Tanda strip (*hyphen*) |
| ® | 00AE | 174 | Tanda merk terdaftar |
| ¯ | 00AF | 175 | Macron |
| ° | 00B0 | 176 | Tanda derajat |
| ± | 00B1 | 177 | Tanda kurang lebih (plus-minus) |
| ² | 00B2 | 178 | Tanda kuadrat (pangkat dua) |
| ³ | 00B3 | 179 | Tanda kubik (pangkat tiga) |
| ´ | 00B4 | 180 | Acute accent |
| µ | 00B5 | 181 | Micro sign |
| ¶ | 00B6 | 182 | Pilcrow sign |
| · | 00B7 | 183 | Middle dot |
|  |  |  |  |
|  |  |  |  |

1. Carilah Daftar perintah bahasa assembly untuk mesin keluarga intel x86lengkap (dari buku referensi atau internet). Daftar perintah ini daat digunakan sebagai pedoman untuk memahami program boot.asm dan kernel.asm.

## Jawab: Bahasa Assembly x86

Akhirnya kita berkenalan dengan bahasanya setelah sekian lama berpusing-pusing dengan memori. Di bagian ini **tidak semua perintah assembly** saya jelaskan, hanya yang penting-penting dan sering dipakai saja. Bila ingin mendapatkan daftar perintah assembly, di internet maupun gramedia banyak.

Komentar

Komentar diawali dengan tanda titik koma (;).

; ini adalah komentar

Label

Label diakhiri dengan tanda titik dua (:).

Contoh: main: ,loop: ,proses: ,keluar:

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 kita. 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. Program EXE harus punya stack. Kira-kira yang penting itu dulu. Semua directive yang dikenal assembler adalah: .186 .286 .286c .286p .287 .386 .386c .386p .387 .486 .486p .8086 .8087 .alpha .break .code .const .continue .cref .data .data? .dosseg .else .elseif .endif .endw .err .err1 .err2 .errb .errdef .errdif .errdifi .erre .erridn .erridni .errnb .errndef .errnz .exit .fardata .fardata? .if .lall .lfcond .list .listall .listif .listmacro .listmacroall .model .no87 .nocref .nolist .nolistif .nolistmacro .radix .repeat .sall .seq .sfcond .stack .startup .tfcond .type .until .untilcxz .while .xall .xcref .xlist. Pusing? Nggak usah dipikirin.

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 untuk assembly yang melibatkan bilangan pecahan (floating point) saya belum belajar, jadi dalam tutorial ini saya hanya memakai bilangan bulat (integer). Sebenarnya masih ada 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. Tapi semua ini ribet dan bikin pusing pembaca.

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. Perintah menarik yang tidak dijelaskan di sini: 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.

Pengulangan

**LOOP** : loop. Mengulang sebuah proses. Pertama register CX dikurangi satu. Bila CX sama dengan nol, maka looping berhenti. Bila tidak nol, maka lompat ke label tujuan. sintaks: LOOP {label tujuan} Tips: isi CX dengan nol untuk mendapat jumlah pengulangan terbanyak. Karena nol dikurang satu sama dengan -1, atau dalam notasi *two’s complement* menjadi FFFF(hex) yang sama dengan 65535(dec). **LOOPE** : loop while equal. Melakukan pengulangan selama CX ≠ 0 dan ZF = 1. CX tetap dikurangi 1 sebelum diperiksa. sintaks: LOOP {label tujuan} **LOOPZ** : loop while zero. Identik dengan LOOPE. **LOOPNE** : loop while not equal. Melakukan pengulangan selama CX ≠ 0 dan ZF = 0. CX tetap dikurangi 1 sebelum diperiksa. sintaks: LOOPNE {label tujuan} **LOOPNZ** : loop while not zero. Identik dengan LOOPNE. **REP** : repeat. Mengulang perintah sebanyak CX kali. sintaks: REP {perintah assembly} contoh:

*mov CX, 05 rep inc BX ;register BX ditambah 1 sebanyak 5x.*

**REPE** : repeat while equal. Mengulang perintah sebanyak CX kali, tetapi pengulangan segera dihentikan bila didapati ZF = 1. sintaks: REPE {perintah assembly} **REPZ** : repeat while zero. Identik dengan REPE. **REPNE** : repeat while not equal. Mengulang perintah sebanyak CX kali, tetapi pengulangan segera dihentikan bila didapati ZF = 0. sintaks: REPNE {perintah assembly} **REPNZ** : repeat while not zero. Identik dengan REPNE.

Perbandingan

**CMP** : compare. Membandingkan dua buah operand. Hasilnya mempengaruhi sejumlah flag register. sintaks: CMP {operand 1}, {operand 2}. Operand ini bisa register dengan register , register dengan isi memori, atau register dengan angka. CMP tidak bisa membandingkan isi memori dengan isi memori. Hasilnya adalah:

|  |  |  |  |
| --- | --- | --- | --- |
| Kasus | Bila operand 1 < operand 2 | Bila operand 1 = operand 2 | Bila operand 1 > operand 2 |
| Signed binary | OF = 1, SF = 1, ZF = 0 | OF = 0, SF = 0, ZF = 1 | OF = 0, SF = 0, ZF = 0 |
| Unsigned binary | CF = 1, ZF = 0 | CF = 0, ZF = 1 | CF = 0, ZF = 0 |

Lompat-lompat

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PERINTAH** | **ARTI** | **SYARAT** | **KASUS** | **KETERANGAN (“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 ∨ ZF = 1 | unsigned | lompat bila op 1 ≥ op 2 | ya |
| **JNB** | jump if not below |
| **JBE** | jump if below or equal | CF = 1 ∨ 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 ∨ 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 < op 2 | ya |
| **JNGE** | jump if not greater or equal |
| **JLE** | jump if less or equal | OF = 1 ∨ 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 1 = 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 1 ≠ 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 |
| **JNP** | jump on not parity | PF = 0 | N/A | lompat bila parity flag = 0 | tidak selalu |
| **JPO** | jump on parity odd | lompat bila bilangan ganjil |
| **JO** | jump if overflow | OF = 1 | N/A | lompat bila overflow flag = 1 | tidak |
| **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 |

Operasi stack

**PUSH** : push. Menambahkan sesuatu ke stack. Sesuatu ini harus register berukuran 16 bit (pada 386+ harus 32 bit), tidak boleh angka, tidak boleh alamat memori. Maka Anda tidak bisa mem-push register 8-bit seperti AH, AL, BH, BL, dan kawan-kawannya. sintaks: push {register 16-bit sumber} contoh: push DX push AX Setelah operasi push, register SP (stack pointer) otomatis dikurangi 2 (karena datanya 2 byte). Makanya, “top” dari stack seakan-akan “tumbuh turun”. **POP** : pop. Mengambil sesuatu dari stack. Sesuatu ini akan disimpan di register tujuan dan harus 16-bit. Maka Anda tidak bisa mem-pop menuju AH, AL, dkk. sintaks: POP {register 16-bit tujuan} contoh: POP BX Setelah operasi pop, register SP otomatis ditambah 2 (karena 2 byte), sehingga “top” dari stack “naik” lagi. Tip: karena register segmen tidak bisa diisi langsung nilainya, Anda bisa menggunakan stack sebagai perantaranya. Contoh kodenya: mov AX, seg teks1 push AX pop DS **PUSHF** : push flags. Mem-push **semua** isi register flag ke dalam stack. Biasa dipakai untuk mem*backup* data di register flag sebelum operasi matematika. Sintaks: PUSHF ;(saja). **POPF** : pop flags. Lawan dari pushf. Sintaks: POPF ;(saja).

**POPA** : pop all general-purpose registers. Adalah ringkasan dari sejumlah perintah dengan urutan:

*pop DI pop SI pop BP pop SP pop BX pop DX pop CX pop AX*

Urutan sudah ditetapkan seperti itu. sintaks: POPA ;(saja). Jauh lebih cepat mengetikkan POPA daripada mengetik POP-POP-POP yang banyak itu, bukan?

**PUSHA** : push all general-purpose registers. Lawan dari POPA, dimana PUSHA adalah singkatan dari sejumlah perintah dengan urutan yang sudah ditetapkan:

*push AX push CX push DX push BX push SP push BP push SI push DI*

Jangan tanyakan kenapa setelah urutannya AX langsung CX serta DX dan bukannya BX dulu. Bukan saya yang ngatur urutannya seperti itu.

Operasi pada register flag

**CLC** : clear carry flag. Menjadikan CF = 0. Sintaks: CLC ;(saja). **STC** : set carry flag. Menjadikan CF = 1. Sintaks: STC ;(saja). **CMC** : complement carry flag. Melakukan operasi NOT pada CF. Yang tadinya 0 menjadi 1, dan sebaliknya. **CLD** : clear direction flag. Menjadikan DF = 0. Sintaks: CLD ;(saja). **STD** : set direction flag. Menjadikan DF = 1. **CLI** : clear interrupt flag. Menjadikan IF = 0, sehingga interrupt ke CPU akan di-disable. Biasanya perintah CLI diberikan sebelum menjalankan sebuah proses penting yang riskan gagal bila diganggu. **STI** : set interrupt flag. Menjadikan IF = 1.

Perintah lainnya

**ORG** : origin. Mengatur awal dari program (bagian static data). Analoginya seperti mengatur dimana letak titik (0, 0) pada koordinat Cartesius. sintaks: ORG {alamat awal} Pada program COM (program yang berekstensi .com), harus ditulis “ORG 100h” untuk mengatur alamat mulai dari progam pada 0100(hex), karena dari alamat 0000(hex) sampai 00FF(hex) sudah dipesan oleh sistem operasi (DOS). **INT** : interrupt. Menginterupsi prosesor. Prosesor akan: 1. Membackup data registernya saat itu, 2. Menghentikan apa yang sedang dikerjakannya, 3. Melompat ke bagian interrupt-handler (entah dimana kita tidak tahu, sudah ditentukan BIOS dan DOS), 4. Melakukan interupsi, 5. Mengembalikan data registernya, 6. Meneruskan pekerjaan yang tadi ditunda. sintaks: INT {nomor interupsi} **IRET** : interrupt-handler return. Kita bisa membuat interrupt-handler sendiri lho! Ada caranya. Perintah IRET adalah perintah yang menandakan bahwa interrupt-handler kita selesai, dan prosesor boleh melanjutkan pekerjaan yang tadi tertunda. **CALL** : call procedure. Memanggil sebuah prosedur. Soal ini saya belum mengerti benar, jadi penjelasannya sementara tidak ada dulu. sintaks: CALL {label nama prosedur} **RET** : return. Tanda selesai prosedur. Setiap prosedur harus memiliki RET di ujungnya. Soal ini saya belum mengerti benar, jadi penjelasannya sementara tidak ada dulu. sintaks: RET ;(saja) **HLT** : halt. Membuat prosesor menjadi tidak aktif. Prosesor harus mendapat interupsi dari luar atau di-reset supaya aktif kembali. **Jadi, jangan gunakan perintah HLT untuk mengakhiri program!!** Sintaks: HLT ;(saja). **NOP** : no operation. Perintah ini memakan 1 byte di memori tetapi tidak menyuruh prosesor melakukan apa-apa selama 3 clock prosesor. Berikut contoh potongan program untuk melakukan *delay* selama 0,1 detik pada prosesor Intel 80386 yang berkecepatan 16 MHz.

*mov ECX, 533333334d ;ini adalah bilangan desimal idle: nop loop idle*

Program assembly pertama

Coba ketik kode berikut di notepad++.

.model small

.code

org 100h

main:

mov AH, 02

mov DL, 65

int 21h

int 20h

end main

Simpan sebagai ‘HurufA.asm’. Kemudian assemble file tadi dengan assembler, misalnya TASM (turbo assembler). Bila bingung cari dimana, silakan download di [sini](http://s.itb.ac.id/~yusufat/TASM.zip). Lebih bagus lagi bila Anda menjalankannya di sistem operasi DOS (seperti MS DOS 6.22 dalam Virtual PC). Cara assemble-nya adalah tasm namafile.asm.

E:\> tasm hurufa.asm

Turbo Assembler Version 2.0 Copyright (c) 1988, 1990 Borland International

Assembling file: hurufa.asm

Error messages: None

Warning messages: None

Passes: 1

Remaining memory: 405k

Kemudian link file tadi dengan cara tlink namafile.obj untuk menjadikannya EXE atau tlink /t namafile.obj untuk menjadikan COM.

E:\> tlink /t hurufa.obj

Turbo Link Version 3.0 Copyright (c) 1987, 1990 Borland International

Karena program kita tidak memiliki direktif .stack dan ada tertulis org 100h, maka program kita harus menjadi COM, bukan EXE. Maka dari itu, jangan melakukan linking tanpa parameter ‘/t’. Kemudian coba jalankan program itu.

E:\> hurufa.com

A

E:\> \_