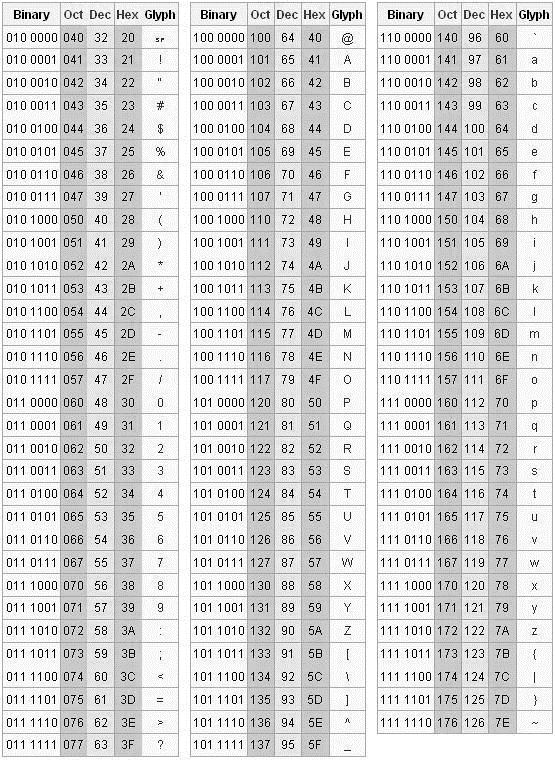
**Tugas Ebook Modul 1**

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, binary dan hexadesimal serta karakter dan simbol yang dikodekan.

Jawab :

ASCII (American Standard Code for Information Interchange) merupakan Kode Standar Amerika untuk Pertukaran Informasi atau sebuah standar internasional dalam pengkodean huruf dan simbol seperti Unicode dan Hex tetapi ASCII lebih bersifat universal.

Tabel ASCII :



1. Carilah daftar perintah bahasa assembly 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’.

|  |  |  |
| --- | --- | --- |
| **N**  **o** | **Perintah** | **Penjelasan** |
| 1. | Komentar | Komentar diawali dengan tanda titik koma (;). |
| 2. | Label | Label diakhiri dengan tanda titik dua (:) |
| 3. | Assembler directives | |
| a. Directives | perintah yang ditujukan kepada assembler ketika sedang menerjemahkan program kita ke bahasa mesin. Directive dimulai dengan tanda titik |
| b. .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. |
| c. .data | memberitahu assembler bahwa bagian di bawah ini adalah data program. |
| d. .code | memberitahu assembler bahwa bagian di bawah ini adalah data program. |
| e. .stack | memberitahu assembler bahwa program kita memiliki stack. |
| 4. | Definisi data | |
| a. DB : Define  Bytes | Membentuk data byte demi byte. Data bisa data numerik maupun teks. |
| b. DW :  Define Words | Membentuk data word demi word (1 word = 2 byte). |
| c. DD : Define  Double  Words | Membentuk data doubleword demi doubleword (4 byte). |
| d. EQU :  Equals | Membentuk konstanta. |
| 5. | Perpindahan data | |

|  |  |  |
| --- | --- | --- |
|  | a. MOV :  Move | Memindahkan suatu nilai dari register ke memori, memori ke register, atau register ke register. |
| b. LEA : Load  Effective  Address | Mengisi suatu register dengan alamat offset sebuah data. |
| c. XCHG :  Exchange | Menukar dua buah register langsung. |
| d. Lainnya : | IN, OUT, LODS, LODSB, LODSW, MOVS, MOVSB, MOVSW, LDS, LES, LAHF, SAHF, dan XLAT. |
| 6. | Operasi logika | |
| a. AND | melakukan bitwise AND. |
| b. OR | melakukan bitwise OR. |
| c. NOT | Melakukan bitwise NOT. |
| d. XOR | Melakukan bitwise XOR. |
| e. SHL : shift  left | Menggeser bit ke kiri. Bit paling kanan diisi nol. |
| f. SHR : shift  right | Menggeser bit ke kanan. Bit paling kiri diisi nol. |
| g. ROL : rotate  left | Memutar bit ke kiri. Bit paling kiri jadi paling kanan kali ini. |
| h. ROR:rotate  right | Memutar bit ke kanan. Bit paling kanan jadi paling kiri. |
| i. Lainnya | RCL dan RCR. |
| 7. | Operasi matematika | |
| a. ADD : Add | Menjumlahkan dua buah register. |
| b. ADC : Add  With Carry | Menjumlahkan dua register dan carry flag (CF). |
| c. INC :  Increment | Menjumlah isi sebuah register dengan 1. |
| d. SUB :  Substract | Mengurangkan dua buah register. |

|  |  |  |
| --- | --- | --- |
|  | e. SBB :  Substract  With  Borrow | Mengurangkan dua register dan carry flag (CF). |
| f. DEC:Decre ment. | Mengurang isi sebuah register dengan 1. |
| g. MUL :  Multiply. | Mengalikan register dengan AX atau AH. |
| h. IMUL :  Signed  Multiply. | Sama dengan MUL, hanya saja IMUL menganggap bit-bit yang ada di register sumber sudah dalam bentuk two’s complement. |
| i. DIV : Divide | Membagi AX atau DX:AX dengan sebuah register. |
| j. IDIV: Signed  Divide | Sama dengan DIV, hanya saja IDIV menganggap bit-bit yang ada di register sumber sudah dalam bentuk two’s complement. |
| k. NEG :  Negate | Membuat isi register menjadi negatif (two’s complement). |
| 8. | Pengulangan | |
| a. 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. |
| b. LOOPE :  Loop While  Equal | Melakukan pengulangan selama CX ≠ 0 dan ZF = 1. CX tetap dikurangi 1 sebelum diperiksa. |
| c. LOOPZ :  Loop While  Zero | Identik dengan LOOPE. |
| d. LOOPNE : Loop While  Not Equal | Melakukan pengulangan selama CX ≠ 0 dan ZF = 0. CX tetap dikurangi 1 sebelum diperiksa. |

|  |  |  |
| --- | --- | --- |
|  | e. LOOPNZ : Loop While  Not Zero | Identik dengan LOOPNE. |
| f. REP : repeat | Mengulang perintah sebanyak CX kali. |
| g. REPE :  Repeat  While Equal | Mengulang perintah sebanyak CX kali, tetapi pengulangan segera dihentikan bila didapati ZF = 1 |
| h. REPZ :  Repeat  While Zero | Identik dengan REPE. |
| i. REPNE :  Repeat  While Not  Equal | Mengulang perintah sebanyak CX kali, tetapi pengulangan segera dihentikan bila didapati ZF = 0. |
| j. REPNZ :  Repeat  While Not  Zero | Identik dengan REPNE |
| 9. | Perbandingan | |
| a. CMP :  Compare | Membandingkan dua buah operand. Hasilnya mempengaruhi sejumlah flag register. |
| 10. | Lompat-lompat |  |
| a. JMP: Jump | Lompat tanpa syarat. Lompat begitu saja. |
| b. Lompat  bersyarat | Sintaksnya sama dengan JMP, yaitu perintah jump diikuti label tujuan. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 11. | Operasi stack |  |
| a. PUSH : Push | Menambahkan sesuatu ke stack. |
| b. POP : Pop | Mengambil sesuatu dari stack. |
| c. PUSHF :  Push Flags | Mem-push semua isi register flag ke dalam stack. Biasa dipakai untuk membackup data di register flag sebelum operasi matematika. |
| d. POPF : Pop  Flags | Lawan dari pushf. |
| e. 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. |

|  |  |  |
| --- | --- | --- |
|  | f. PUSHA : Push All  GeneralPurpose  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 |
| 12. | Operasi pada register flag | |
| a. CLC : Clear  Carry Flag | Menjadikan CF = 0. |
| b. STC : Set  Carry Flag | Menjadikan CF = 1. |
| c. CMC:  Complement  Carry Flag | Melakukan operasi NOT pada CF. Yang tadinya 0 menjadi 1, dan sebaliknya. |
| d. CLD : Clear  Direction  Flag | Menjadikan DF = 0. |
| e. STD : Set  Direction  Flag | Menjadikan DF = 1. |
| f. 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 |
| g. STI : Set  Interrupt  Flag | Menjadikan IF = 1. |
| 13. | Perintah lainnya | |
| a. ORG :  Origin | Mengatur awal dari program (bagian static data). Analoginya seperti mengatur dimana letak titik (0, 0) pada koordinat Cartesius. |
|  | b. 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. |
| c. IRET :  InterruptHandler Return. | Dapat membuat interrupt-handler sendiri dengan berbagai cara. Perintah IRET adalah perintah yang menandakan bahwa interrupt-handler kita selesai, dan prosesor boleh melanjutkan pekerjaan yang tadi tertunda. |
| d. CALL : Call  Procedure | Memanggil sebuah prosedur. |
| e. RET :  Return | Tanda selesai prosedur.  Setiap prosedur harus memiliki RET di ujungnya. |
| f. HLT : Halt | Membuat prosesor menjadi tidak aktif.  Prosesor harus mendapat interupsi dari luar atau di-reset supaya aktif kembali. Jangan gunakan perintah HLT untuk mengakhiri program. |
| g. NOP : No  Operation | Perintah ini memakan 1 byte di memori tetapi tidak menyuruh prosesor melakukan apa-apa selama 3 clock prosesor. |