TUGAS PRAKTIKUM PERTEMUAN PERTAMA

Nama : W. Faisal Hari Dewanto

NIM : L200180046

Kelas : B

1. Apa yang dimaksud dengan kode ‘ASCII’, buatlah table kode ASCII lengkap cukup kode ASCII yang stadar tidak perlu extended, tuliskan kode ASCII dalam format angka decimal, binary, dan hexadecimal 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 dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 7 bit. Namun, ASCII disimpan sebagai sandi 8 bit dengan menambakan satu angka 0 sebagai bit significant paling tinggi. Bit tambahan ini sering digunakan untuk uji paritas. Karakter control pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi logical communication, Device control, Information separator, Code extention, dan physical communication. Code ASCII ini banyak dijumpai pada papan ketik (keyboard) computer atau instrument-instrument digital.

TABEL ASCII:

|  |  |  |  |
| --- | --- | --- | --- |
| Nilai ANSI ASCII (Desimal) | Nilai Unicode (Heksa Desimal) | Binner | Karakter |
| 0 | 00 | 00000000 | NUL |
| 1 | 01 | 00000001 | SOH |
| 2 | 02 | 00000010 | STX |
| 3 | 03 | 00000011 | ETX |
| 4 | 04 | 00000100 | EOT |
| 5 | 05 | 00000101 | ENQ |
| 6 | 06 | 00000110 | ACK |
| 7 | 07 | 00000111 | BEL |
| 8 | 08 | 00001000 | BS |
| 9 | 09 | 00001001 | HT |
| 10 | 0A | 00001010 | LF |
| 11 | 0B | 00001011 | VT |
| 12 | 0C | 00001100 | FF |
| 13 | 0D | 00001101 | CR |
| 14 | 0E | 00001110 | SO |
| 15 | 0F | 00001111 | SI |
| 16 | 10 | 00010000 | DLE |
| 17 | 11 | 00010001 | DC1 |
| 18 | 12 | 00010010 | DC2 |
| 19 | 13 | 00010011 | DC3 |
| 20 | 14 | 00010100 | DC4 |
| 21 | 15 | 00010101 | NAK |
| 22 | 16 | 00010110 | SYN |
| 23 | 17 | 00010111 | ETB |

|  |  |  |  |
| --- | --- | --- | --- |
| 24 | 18 | 00011000 | CAN |
| 25 | 19 | 00011001 | EM |
| 26 | 1A | 00011010 | SUB |
| 27 | 1B | 00011011 | ESC |
| 28 | 1C | 00011100 | FS |
| 29 | 1D | 00011101 | GS |

|  |  |  |  |
| --- | --- | --- | --- |
| 30 | 1E | 00011110 | RS |
| 31 | 1F | 00011111 | US |
| 32 | 20 | 00100000 | space |
| 33 | 21 | 00100001 | ! |
| 34 | 22 | 00100010 | " |
| 35 | 23 | 00100011 | # |
| 36 | 24 | 00100100 | $ |
| 37 | 25 | 00100101 | % |
| 38 | 26 | 00100110 | & |
| 39 | 27 | 00100111 | ' |
| 40 | 28 | 00101000 | ( |
| 41 | 29 | 00101001 | ) |
| 42 | 2A | 00101010 | \* |
| 43 | 2B | 00101011 | + |
| 44 | 2C | 00101100 | , |
| 45 | 2D | 00101101 | - |
| 46 | 2E | 00101110 | . |
| 47 | 2F | 00101111 | / |
| 48 | 30 | 00110000 | 0 |
| 49 | 31 | 00110001 | 1 |
| 50 | 32 | 00110010 | 2 |
| 51 | 33 | 00110011 | 3 |
| 52 | 34 | 00110100 | 4 |
| 53 | 35 | 00110101 | 5 |
| 54 | 36 | 00110110 | 6 |
| 55 | 37 | 00110111 | 7 |
| 56 | 38 | 00111000 | 8 |
| 57 | 39 | 00111001 | 9 |
| 58 | 3A | 00111010 | : |
| 59 | 3B | 00111011 | ; |
| 60 | 3C | 00111100 | < |
| 61 | 3D | 00111101 | = |
| 62 | 3E | 00111110 | > |
| 63 | 3F | 00111111 | ? |
| 64 | 40 | 01000000 | @ |
| 65 | 41 | 01000001 | A |
| 66 | 42 | 01000010 | B |
| 67 | 43 | 01000011 | C |
| 68 | 44 | 01000100 | D |
| 69 | 45 | 01000101 | E |
| 70 | 46 | 01000110 | F |
| 71 | 47 | 01000111 | G |
| 72 | 48 | 01001000 | H |
| 73 | 49 | 01001001 | I |
| 74 | 4A | 01001010 | J |

|  |  |  |  |
| --- | --- | --- | --- |
| 75 | 4B | 01001011 | K |
| 76 | 4C | 01001100 | L |
| 77 | 4D | 01001101 | M |
| 78 | 4E | 01001110 | N |
| 79 | 4F | 01001111 | O |
| 80 | 50 | 01010000 | P |
| 81 | 51 | 01010001 | Q |
| 82 | 52 | 01010010 | R |
| 83 | 53 | 01010011 | S |
| 84 | 54 | 01010100 | T |
| 85 | 55 | 01010101 | U |
| 86 | 56 | 01010110 | V |
| 87 | 57 | 01010111 | W |
| 88 | 58 | 01011000 | X |
| 89 | 59 | 01011001 | Y |
| 90 | 5A | 01011010 | Z |
| 91 | 5B | 01011011 | [ |
| 92 | 5C | 01011100 | \ |
| 93 | 5D | 01011101 | ] |
| 94 | 5E | 01011110 | ^ |
| 95 | 5F | 01011111 | \_ |
| 96 | 60 | 01100000 | ` |
| 97 | 61 | 01100001 | a |
| 98 | 62 | 01100010 | b |
| 99 | 63 | 01100011 | c |
| 100 | 64 | 01100100 | d |
| 101 | 65 | 01100101 | e |
| 102 | 66 | 01100110 | f |
| 103 | 67 | 01100111 | g |
| 104 | 68 | 01101000 | h |
| 105 | 69 | 01101001 | i |
| 106 | 6A | 01101010 | j |
| 107 | 6B | 01101011 | k |
| 108 | 6C | 01101100 | l |
| 109 | 6D | 01101101 | m |
| 110 | 6E | 01101110 | n |
| 111 | 6F | 01101111 | o |
| 112 | 70 | 01110000 | p |
| 113 | 71 | 01110001 | q |
| 114 | 72 | 01110010 | r |
| 115 | 73 | 01110011 | s |
| 116 | 74 | 01110100 | t |
| 117 | 75 | 01110101 | u |
| 118 | 76 | 01110110 | v |
| 119 | 77 | 01110111 | w |

|  |  |  |  |
| --- | --- | --- | --- |
| 120 | 78 | 01111000 | x |
| 121 | 79 | 01111001 | y |
| 122 | 7A | 01111010 | z |
| 123 | 7B | 01111011 | { |
| 124 | 7C | 01111100 | | |
| 125 | 7D | 01111101 | } |
| 126 | 7E | 01111110 | ~ |
| 127 | 7F | 01111111 | DEL |

Binary Oct Dec Hex Gtjpll

100 0000 100 64 40 @

"

1 00 0001 101 65 41 A

8

|  |  |  |  |
| --- | --- | --- | --- |
| 100 0010 102 | | 66 | 42 |
| 100 0011 | 103 | 67 | 43 |
| 100 0100 | 104 | 68 | 44 |
| 100 0101 | 105 | 69 | 45 |
| 1000110 | 106 | 70 | 46 |
| 100 0111 107 | | 71 | 47 |
| 100 1000 | 110 | 72 | 48 |
| 1001001 | 111 | 73 | 49 |
| 1001010 | 112 | 74 | 4A |
| 1001011113 | | 75 | 48 |
| 1001100 | 114 | 76 | 4C |
| 100 1101 | 115 | *77* | 40 |
| 1001110 | 116 |  | 4E |
| 1001111 | 117 | 79 | 4F |
| 101 0000 | 120 | 80 | 50 |
| 101 0001 | 121 | 81 | 51 |
| 101 0010 |  | f--l |  |
| 122 82 | | 52 |
| 101 0011 |  | +----! |  |
| 123 83 | | 53 |
| 101 0100 |  | >-< |  |
| 124 | 84 | 54 |
| 101 0101 | 125 | 85 | 55 |
| 101 0110 | 126 | 86 | 56 |
| 101 0111 | 127 | 87 | 57 |
| 1011000 130 | | 88 | 58 |
| 1011001 131 | | 89 | 59 |
| 1011010 132 | | 90 | 5A |
| 1011011 | 133 | 91 | 58 |
| 1011100 | 134 | 92 | SC |
| 1011101 | 135 | 93 | 50 |
| 1011110 | 136 | 94 | 5E |
| 1011111 | 137 | 95 | 5F |

c

0

E

%

F

- 0

•

H

J

|  |  |
| --- | --- |
| K | |
| L | |
|  | |
|  | N |
|  | 0  p |
|  | Q |
|  | R |
|  | s |
|  | T |
| I | u  *v* |

"'

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Binaiy | Oct | Dec | Hex | Glyt>ll |
| 1100000 140 | | 96 | 60 |  |
| 1100001 141 | | 97 | 61 | a |
| ,1100010.142 | | 98 | 62 | b |
| 1100011 143 99 | | | 63 | c |
| !1100100 144 100 | | | 64 | d |
| 1100101145101 | | | 65 | e |
| 1100110 | 146 | 102 | 66 | r |
| 1100111 | 147 | 103 | 67 | g |
| 1101000 | 150 | 104 | 68 | h |
| 1101001 | 151 | 105 | 69 |  |
| 1101010 | 152 | 106 | 6A |  |
| 1101011153107 | | | 68 | k |
| 11011oor154 | | 108 | 6C |  |
| 1101101 r155 | | 109 | 60 | m |
|  |  |  | 6E | n |
| 1101111'157 | | 111 | 6F | 0 |
| 1110000160 | | 112 | 70 | p |
| 111 0001 161 | | 113 | 71 | q |
| 1110010162 | | 114 | 72 | r |
| 1111 0011.163 | | 115 | 73 | s |
|  | 164 | 116 | 74 |  |
| 111 0101 165 117 | | | 75 | u |
| 11110110166 118 | | | 76 | v |
| 111 0111 167 | | 119 | *77* | w |
| �1000 170 120 | | | 78 | x |
| 1111001 171 121 | | | 79 | y |
| 1111010 172 122 | | | 7A | z |
| 1111011 173 | | 123 | 78 | { |
| 1111100 174 | | 124 | 7C | I |
| 1111101 175 | | 125 | 70 | ) |
| 1  11111110 115 126 | | | 7E |  |



066 5-4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bina1y | Oct Dec | | Hex Glypt1 | |
| 010 0000 | 040 32 | | 20 |  |
| 010 0001 | 041 | f--  33  f-- | 21  22  23  24  25  26  27  28  29  2A  28  2C  20  2E  2F  30  31  32  33  34  35  36  37  38  39  3A  38  3C  30  3E  3F | ! |
| 010 0010 | .. |
| 042 34 | |
| 010 0011  -010 -0100  010 0101 | 043  044  045  046  047  050  051  052 | -  35  36-  37  38  39  40  41  42 | #  $ |
| -  &-  ( |
| 010 0110  01 0 0111  010 1000  010 1001  010 1010 |
| )  • |
| 01 0 1 011 | 053 43 | | + |
| 010 1100 |  | ,----- | • |
| 054 44 | |
| 01 0 11 01 |  | I- | - |
| 055 45 | |
| 01 0 111 0 |  | i- |  |
| 056 46 | |
| 01 0 1111  011 0000 |  | f-- | *I*  0 |
| 057 47  060 48 | |
| 011 0001 | 061 | '-  49 | 1 |
| 011 0010 | 062 50 | | 2 |
| 011 0011 | 063 51 | | 3 |
| 011 0100 |  | ,----- | 4 |
| 064 52 | |
| 011 01 01  011 011 0 |  | '-- | 5  6 |
| 065 53 | |
| -011 0111-  011 1000  011 1 001  011 1 01 0  011 1 011  011 11 00 | 067  070  071  072  073  074  075  076 | 55  56  57  58  59  60-  6-1  62 | *7* |
| 8 -  9  -  .-  < |
| 011 11 01 | = |
| 011 111 0 | > |
| 0111111 | 077 63 | | ? |

w

x

y

z

I

\

A

2. Carilah daftar perintah Bahasa assembly untuk mesin intel keluarga x86 lengkap (dari buku referensi atau internet). Daftar perintah ini dapat di gunakan sebagai pedoman untuk memahami program

‘boot.asm’ dan ‘kernel.asm’

Jawab :

Dalam program bahasa assembly terdapat 2 jenis yang kita tulis dalam program:

1. Assembly Directive (yaitu merupakan kode yang menjadi arahan bagi assembler/compiler untuk menata program)

2. Instruksi (yaitu kode yang harus dieksekusi oleh CPU mikrokontroler dengan melakukan operasi tertentu sesuai dengan daftar yang sudah tertanam dalam CPU)

Daftar Assembly Directive

Assembly Directive Keterangan

EQU Pendefinisian konstanta

DB Pendefinisian data dengan ukuran satuan 1 byte DW Pendefinisian data dengan ukuran satuan 1 word DBIT Pendefinisian data dengan ukuran satuan 1 bit DS Pemesanan tempat penyimpanan data di RAM ORG Inisialisasi alamat mulai program

END Penanda akhir program

CSEG Penanda penempatan di code segment

XSEG Penanda penempatan di external data segment DSEG Penanda penempatan di internal direct data segment ISEG Penanda penempatan di internal indirect data segment BSEG Penanda penempatan di bit data segment

CODE Penanda mulai pendefinisian program

XDATA Pendefinisian external data DATA Pendefinisian internal direct data IDATA Pendefinisian internal indirect data BIT Pendefinisian data bit

#INCLUDE Mengikutsertakan file program lain

Daftar Perintah Bahasa Assembly:

|  |  |
| --- | --- |
| Instruksi | Keterangan Singkatan |
| ACALL | Absolute Call |
| ADD | Add |
| ADDC | Add with Carry |
| AJMP | Absolute Jump |
| ANL | AND Logic |
| CJNE | Compare and Jump if Not Equal |
| CLR | Clear |
| CPL | Complement |
| DA | Decimal Adjust |
| DEC | Decrement |
| DIV | Divide |
| DJNZ | Decrement and Jump if Not Zero |
| INC | Increment |
| JB | Jump if Bit Set |
| JBC | Jump if Bit Set and Clear Bit |
| JC | Jump if Carry Set |
| JMP | Jump to Address |
| JNB | Jump if Not Bit Set |
| JNC | Jump if Carry Not Set |
| JNZ | Jump if Accumulator Not Zero |

|  |  |
| --- | --- |
| JZ | Jump if Accumulator Zero |
| LCALL | Long Call |
| LJMP | Long Jump |
| MOV | Move from Memory |
| MOVC | Move from Code Memory |
| MOVX | Move from Extended Memory |
| MUL | Multiply |
| NOP | No Operation |
| ORL | OR Logic |
| POP | Pop Value From Stack |
| PUSH | Push Value Onto Stack |
| RET | Return From Subroutine |
| RETI | Return From Interrupt |
| RL | Rotate Left |
| RLC | Rotate Left through Carry |

|  |  |
| --- | --- |
| RR | Rotate Right |
| RRC | Rotate Right through Carry |
| SETB | Set Bit |
| SJMP | Short Jump |
| SUBB | Subtract With Borrow |
| SWAP | Swap Nibbles |
| XCH | Exchange Bytes |
| XCHD | Exchange Digits |
| XRL | Exclusive OR Logic |

Untuk yang lebih jelas dan detail:

a. MOV

Perintah MOV adalah perintah untuk mengisi, memindahkan,memperbaruhi isi suatu register, variable ataupun lokasi memory, Adapun tata penulisan perintah MOV adalah :

MOV [operand A], [Operand B] Contoh :

MOV AH,02

Operand A adalah Register AH Operand B adalah bilangan 02

Hal yang dilakukan oleh komputer untuk perintah diatas adalahmemasukan 02 ke register AH.

b. INT (Interrupt)

Bila anda pernah belajar BASIC, maka pasti anda tidak asing lagi dengan perintah GOSUB. Perintah INT juga mempunyai cara kerja yang sama dengan GOSUB, hanya saja subroutine yang dipanggil telah disediakan oleh memory komputer yang terdiri 2 jenis yaitu :

- Bios Interrupt ( interput yang disediakan oleh BIOS (INT 0 – INT 1F))

- Dos Interrupt ( Interrupt yang disediakan oleh DOS (INT 1F – keatas))

c. Push

Adalah perintah untuk memasukan isi register pada stack, dengan tata penulisannya:POP [operand 16 bit]

d. Pop

perintah yang berguna untuk mengeluarkan isi dari register/variable dari stack,dengan tata penulisannya adalah : POP [operand 16 bit]

e. RIP (Register IP)

Perintah ini digunakan untuk memberitahu komputer untuk memulai memproses program dari titik tertentu.

f. A (Assembler)

Perintah Assembler berguna untuk tempat menulis program Assembler.

-A100

0FD8:100

g. RCX (Register CX)

Perintah ini digunakan untuk mengetahui dan memperbaruhi isi register CX yang merupakan tempat penampungan panjang program yang sedang aktif pun, ada yang demikian:

1. Definisi Stack

Secara harfiah stack berarti tumpukan, yaitu bagian memori yang digunakan untuk menyimpan nilai suatu register untuk sementara, membentuk tumpukan nilai. Stack dapat dibayangkan sebagai tabung memanjang (seperti

tabung penyimpan koin). Sedangkan nilai suatu register dapat dibayangkan sebagai koin yang dapat dimasukkan dalam tabung tersebut. Jika ada data yang disimpan maka data-data tersebut akan bergeser ke arah memori rendah, dan akan bergeser kembali ke arah memori tinggi bila data yang disimpan telah diambil.

2. Perintah Perpindahan Data

Terkait perpindahan data, bahasa assembler mempunyai beberapa perintah yang dapat dibedakan yaitu untuk memindahkan data tunggal seperti huruf atau angka dan untuk memindahkan data string yang berupa deretan huruf. Tetapi di sini hanya akan menjelaskan beberapa perintah yang dipakai dalam aplikasi.

2.1. PUSH/POP Syntax :

PUSH Reg16Bit

POP Reg16Bit

PUSH adalah perintah penyimpanan data ke memori stack secara langsung, dan untuk mengambil keluar nilai yang disimpan tersebut gunakan perintah POP. Nilai terakhir yang dimasukkan dalam stack, dengan perintah PUSH, akan terletak pada puncak tabung stack. Dan perintah POP pertama kali akan mengambil nilai pada stack yang paling atas kemudian nilai berikutnya, demikian seterusnya. Jadi nilai yang terakhir dimasukkan akan merupakan yang pertama dikeluarkan. Operasi ini dinamakan LIFO (Last In First Out). Perhatikan contoh berikut ini:

push ax; push bx; push cx;

mov ax, $31C; mov bx; $31D; mov cx, $31E; pop cx;

pop bx;

pop ax;

2.2. MOV Syntax :

MOV destination, source

Digunakan untuk menyalin data dari memori/register ke memori/register atau dari data langsung ke register. Nilai pada source yang dipindahkan tidaklah berubah. Pada contoh di bawah, register al diberi nilai $31C kemudian nilai register al disalin ke register ax. Jadi sekarang nilai register al dan register ax adalah $31C.

mov al, $31C;

mov ax, al;

Hal-hal yang tidak boleh dilakukan dalam penyalinan data:

a. Penyalinan data antarregister segmen (ds, es, cs, ss)

mov ds, es ? tidak dibenarkan

Gunakan register general, misalnya register ax, sebagai perantara mov ax, es

mov ds, ax

atau gunakan stack sebagai perantara push es

pop ds

b. Penyalinan data secara langsung untuk register segmen (ds, es, cs, ss)

mov ds, $31C ? tidak dibenarkan

Gunakan register general, misalnya register ax, sebagai perantara mov ax, $31C

mov ds, ax

c. Penyalinan data langsung antarmemori mov memB, memA ? tidak dibenarkan

Gunakan register general, misalnya register ax, sebagai perantara

mov ax, memA

mov memB, ax

d. Penyalinan data antarregister general yang berbeda daya tampungnya (8 bit dengan 16 bit) tanpa pointer mov al, bx ? tidak dibenarkan

2.3. IN/OUT Syntax :

IN Reg16Bit, port

OUT port, Reg16Bit

Untuk membaca data dari suatu port dan memasukkan nilainya ke dalam suatu register gunakan perintah IN. Dan perintah OUT digunakan untuk memasukkan suatu nilai ke dalam suatu port. Nilai yang akan dimasukkan diberikan pada register al/ax dan alamat port diberikan pada register dx. Pada contoh berikut ini, pertama kali register dx disimpan pada stack, menyalin nilai $31E pada register dx kemudian perintah IN akan membaca nilai pada register dx (port bernilai $31E) dan memasukkannya ke dalam register al. Dan terakhir nilai tersebut disalin ke variabel Data.

push dx

mov dx, $31E

in al, dx

mov Data, al pop dx

Dan contoh berikut untuk memberi nilai ($8A) pada suatu port. push dx

mov dx, $31E mov al, $8A out dx, al

pop dx

3. Operasi Aritmatika

3.1. Penjumlahan

Syntax :

ADD destination, source ADC destination, source INC destination

Perintah ADD akan menjumlahkan nilai pada destination dan source tanpa menggunakan carry (ADD), dimana hasil yang didapat akan ditaruh pada destination. Dalam bahasa pascal pernyataan ini sama dengan pernyataan destination := destination + source. Daya tampung destination dan source harus sama misalnya register al (8 bit) dan ah (8 bit), ax (16 bit) dan bx (16 bit). Perhatikan contoh berikut, nilai register ah sekarang menjadi $10 : mov ah, $5;

mov al, $8;

add ah, al

Perintah ADC digunakan untuk menangani penjumlahan dengan hasil yang melebihi daya tampung destination yaitu dengan menggunakan carry (ADD), dalam bahasa pascal sama dengan pernyataan destination := destination

+ source + carry. Misalnya register ax (daya tampung 16 bit) diberi nilai $1234 dan bx (16 bit) diberi nilai $F221, penjumlahan kedua register ini adalah $10455. Jadi ada bit ke 17 padahal daya tampung register bx hanya 16 bit, penyelesaiannya adalah nilai bx = $0455 dengan carry flag = 1.

Perintah INC digunakan untuk operasi penjumlahan dengan nilai 1. Jadi nilai pada destination akan ditambah 1, seperti perintah destination := destination + 1 dalam bahasa Pascal.

3.2. Pengurangan

Syntax :

SUB destination, source SBB destination, source DEC destination

Perintah SUB untuk mengurangkan 2 operand tanpa carry flag. Hasilnya diletakkan pada destination dalam bahasa pasca sama dengan pernyataan destination := destination – source. Untuk mengenolkan suatu register, kurangkan dengan dirinya sendiri seperti contoh berikut ini. Pertama kali register ax bernilai $5, kemudian nilai register tersebut dikurangi dengan dirinya sendiri sehingga terakhir nilai register ax adalah 0.

mov ax, $15; mov bx, $10; sub ax, bx; sub ax, ax;

Perintah SBB mengurangkan nilai destination dengan nilai source kemudian dikurangi lagi dengan carry flag

(destination := destination – source – carry flag).

Dan perintah DEC untuk mengurangi nilai destination dengan 1.

3.3. Perkalian

Syntax : MUL source

Digunakan untuk mengalikan data pada accumulator dengan suatu operand dan hasilnya diletak pada register source. Register source dapat berupa suatu register 8 bit (misal bl, bh, dan sebagainya), register 16 bit (bx, dx, dan sebagainya) atau suatu variabel.

3.4. Pembagian

Syntax : DIV source

Operasi aritmatika ini pada dasarnya sama dengan operasi perkalian.