Nama : Hanifah Elvira Sukma Dewi

Nim : L200180124

Modul 1 (Tugas)

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, contohnya 124 adalah untuk karakter " | " . Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Sedangkan fungsi dari kode ASCII ialah digunakan untuk mewakili karakter-karakter angka maupun huruf didalam komputer, sebagai contoh dapat kita lihat pada karakter 1, 2, 3, A, B, C, dan sebagainya.

Table ASCII

Dec	Hex	Oct	Char	Binary
0	00	000	NUL	0000 0000
1	01	001	SOH	0000 0001
2	02	002	STX	0000 0010
3	03	003	ETX	0000 0011
4	04	004	EOT	0000 0100
5	05	005	ENQ	0000 0101
6	06	006	ACK	0000 0110
7	07	007	BEL	0000 0111
8	08	008	BS	0000 1000
9	09	009	HT	0000 1001
10	0A	010	LF	0000 1010
11	0B	011	VT	0000 1011
12	0C	012	FF	0000 1100
13	0D	013	CR	0000 1101
14	0E	014	CO	0000 1110

15	0F	015	SI	0000 1111

Daftar perintah assembly untuk mesin intel keluarga x86

Petunjuk	Berarti	Catatan	Opcode
AAA	ASCII menyesuaikan AL setelah penambahan	digunakan dengan decimal kode biner yang dibongkar	0x37
AAD	ASCII menyesuaikan AXE sebelum pembagian	8086/8088 lembar data hanya mendasarkan versi 10 dari instruksi AAD (opcode 0xD5 0x0A), tetapi basis lainnya akan berfungsi. Kemudian, dokumentasi Intel juga memiliki bentuk generik. NEC V20 dan V30 (dan mungkin juga CPU seri NEC lainnya) selalu menggunakan basis 10, dan mengabaikan argumen, menyebabkan sejumlah ketidakcocokan	0xD5
AAM	ASCII menyesuaikan AX setelah perkalian	Hanya versi dasar 10 (Operand 0xA) yang didokumentasikan, lihat catatan untuk AAD	0xD4
AAS	ASCII menyesuaikan AL setelah dikurangi		0x3F
ADC	Tambahkan dengan membawa	destination := destination + source + carry_flag	0x10 ... 0x15, 0x80 / 2 ... 0x83 / 2
MENAMBAHKAN	Menambahkan	(1) r/m += r/imm; (2) r += m/imm;	0x00 ... 0x05, 0x80 / 0 ...

			0x83 / 0
DAN	Logis dan	(1) r/m &= r/imm; (2) r &= m/imm;	0x20 ... 0x25, 0x80 / 4 ... 0x83 / 4
PANGGILAN	Prosedur panggilan	push eip *; eip points to the instruction directly after the call*	0x9A, 0xE8, 0xFF / 2, 0xFF / 3
CBW	Konversi byte ke kata		0x98
CLC	Hapus carry flag	CF = 0;	0xF8
CLD	Hapus arah bendera	DF = 0;	0xFC
CLI	Hapus bendera interupsi	IF = 0;	0xFA
CMC	Melengkapi bendera pembawa		0xF5
CMP	Bandingkan operan		0x38 ... 0x3D, 0x80 / 7 ... 0x83 / 7
CMPSB	Bandingkan byte dalam memori		0xA6
CMPSW	Bandingkan kata-kata		0xA7
CWD	Konversi kata menjadi doubleword		0x99
DAA	Menyesuaikan desimal AL setelah penambahan	(digunakan dengan desimal kode biner dikemas)	0x27
DAS	Sesuaikan desimal AL setelah dikurangi		0x2F
DEC	Pengurangan oleh 1		0x48 ... 0x4F, 0xFE / 1, 0xFF / 1
DIV	Pembagian tanpa tanda	DX:AX = DX:AX / r/m; menghasilkan DX == remainder	0xF6 / 6, 0xF7 / 6
ESC	Digunakan dengan unit floating-point		0xD8..0xDF
HLT	Masukkan status berhenti		0xF4
IDIV	Perpecahan yang ditandatangani	DX:AX = DX:AX / r/m; menghasilkan DX == remainder	0xF6 / 7, 0xF7 / 7
IMUL	Bertanda multiply	(1) DX:AX = AX *	0x69, 0x6B

		r/m; (2) AX = AL * r/m	(keduanya sejak 80186), 0xF6 / 5, 0xF7 / 5, 0x0FAF (sejak 80386)
DI	Input dari port	(1) AL = port[imm]; (2) AL = port[DX]; (3) AX = port[imm]; (4) AX = port[DX];	0xE4, 0xE5, 0xEC, 0xED
INC	Kenaikan menurut 1		0x40 ... 0x47, 0xFE / 0, 0xFF / 0
INT	Panggilan untuk menyela		0xCC, 0xCD
KE	Panggilan untuk menyela jika meluap		0xCE
IRET	Kembali dari interupsi		0xCF
Jcc	Lompat jika kondisinya	(JA, JAE, JB, JBE, JC, JE, JG, JGE, JL, JLE, JNA, JNAE, JNB, JNBE, JNC, JNE, JNG, JNGE, JNL, JNLE, JNO, JNP, JNS, JNZ, JO , JP, JPE, JPO, JS, JZ)	0x70 ... 0x7F, 0x0F80 ... 0x0F8F (sejak 80386)
JCXZ	Lompat jika CX nol		0xE3
JMP	Melompat		0xE9 ... 0xEB, 0xFF / 4, 0xFF / 5
LAHF	Muat FLAGS ke register AH		0x9F
LDS	Load pointer menggunakan DS		0xC5
LEA	Muat Alamat yang Efektif		0x8D
LES	Memuat ES dengan pointer		0xC4
MENGUNCI	Tentukan sinyal BUS LOCK #	(untuk multiprosesing)	0xF0
LODSB	Memuat byte string	if (DF==0) AL = * SI ++ ; else AL = * SI -- ;	0xAC
LODSW	Memuat kata string	if (DF == 0) AX = * SI ++ ; else AX = * SI -- ;	0xAD
LOOP / LOOPx	Kontrol putaran	(LOOPE, LOOPNE,	0xE0 ... 0xE2

		LOOPNZ, LOOPZ) if (x && -- CX) goto lbl ;	
MOV	Pindah	menyalin data dari satu lokasi ke lokasi lain, (1) r/m = r; (2) r = r/m;	0xA0 ... 0xA3
MOVSB	Pindahkan byte dari string ke string	jika (DF == 0) * (byte *) DI ++ = * (byte *) SI ++ ; lain * (byte *) DI - = * (byte *) SI - ;	0xA4
MOVSW	Pindahkan kata dari string ke string	jika (DF == 0) * (kata *) DI ++ = * (kata *) SI ++ ; lain * (kata *) DI - = * (kata *) SI - ;	0xA5
MUL	Multiply tanpa tanda	(1) DX:AX = AX * r/m; (2) AX = AL * r/m;	0xF6 / 4 ... 0xF7 / 4
NEG	Negasi komplemen dua	r / m * = - 1 ;	0xF6 / 3 ... 0xF7 / 3
NOP	Tidak ada operasi	opcode setara dengan XCHG EAX, EAX	0x90
TIDAK	Meniadakan operan, TIDAK logis	r / m ^ = - 1 ;	0xF6 / 2 ... 0xF7 / 2
ATAU	Logis atau	(1) r / m | = r / imm ; (2) r | = m / imm ;	0x08 ... 0x0D, 0x80 ... 0x83 / 1
DI LUAR	Output ke port	(1) port[imm] = AL; (2) port[DX] = AL; (3) port[imm] = AX; (4) port[DX] = AX;	0xE6, 0xE7, 0xEE, 0xEF
POP	Pindahkan data dari tumpukan	r/m = * SP++; POP CS (opcode 0x0F) hanya berfungsi pada 8086/8088. CPU kemudian menggunakan	0x07, 0x0F (hanya 8086/8088), 0x17, 0x1F, 0x58… 0x5F,

		0x0F sebagai awalan untuk instruksi yang lebih baru.	0x8F / 0
POPF	BENDERA Pop mendaftar dari tumpukan	FLAGS = * SP++;	0x9D
DORONG	Dorong data ke tumpukan	* -- SP = r / m ;	0x06, 0x0E, 0x16, 0x1E, 0x50 ... 0x57, 0x68, 0x6A (keduanya sejak 80186), 0xFF / 6
PUSHF	Dorong BENDERA ke tumpukan	* -- SP = FLAGS ;	0x9C
RCL	Putar ke kiri (dengan membawa)		0xC0 ... 0xC1 / 2 (sejak 80186), 0xD0 ... 0xD3 / 2
RCR	Putar ke kanan (dengan membawa)		0xC0 ... 0xC1 / 3 (sejak 80186), 0xD0 ... 0xD3 / 3
REPxx	Ulangi MOVS / STOS / CMPS / LODS / SCAS	(REP, REPE, REPNE, REPNZ, REPZ)	0xF2, 0xF3
MEMBASAHI	Kembali dari prosedur	Bukan instruksi nyata. Assembler akan menerjemahkannya ke RETN atau RETF tergantung pada model memori sistem target.	
RETN	Kembali dari prosedur terdekat		0xC2, 0xC3
RETF	Kembali dari prosedur jauh		0xCA, 0xCB
ROL	Putar ke kiri		0xC0 ... 0xC1 / 0 (sejak 80186), 0xD0 ... 0xD3 / 0
ROR	Putar ke kanan		0xC0 ... 0xC1 / 1 (sejak 80186), 0xD0 ... 0xD3 / 1

SAHF	Simpan AH ke dalam BENDERA		0x9E
SAL	Shift Arithmetically left (shift shift kiri)	(1) r/m <<= 1; (2) r/m <<= CL;	0xC0 ... 0xC1 / 4 (sejak 80186), 0xD0 ... 0xD3 / 4
SAR	Bergeser ke kanan secara Aritmatika (bergeser ke kanan)	(1) (signed) r/m >>= 1; (2) (signed) r/m >>= CL;	0xC0 ... 0xC1 / 7 (sejak 80186), 0xD0 ... 0xD3 / 7
SBB	Pengurangan dengan pinjaman	alternatif 1-byte encoding SBB AL, AL tersedia melalui instruksi SALC tidak berdokumen	0x18 ... 0x1D, 0x80 ... 0x83 / 3
SCASB	Bandingkan string byte		0xAE
SCASW	Bandingkan string kata		0xAF
SHL	Shift kiri (shift kiri yang tidak bertanda tangan)		0xC0 ... 0xC1 / 4 (sejak 80186), 0xD0 ... 0xD3 / 4
SHR	Geser ke kanan (geser kanan ke kiri)		0xC0 ... 0xC1 / 5 (sejak 80186), 0xD0 ... 0xD3 / 5
STC	Atur flag carry	CF = 1;	0xF9
STD	Tetapkan bendera arah	DF = 1;	0xFD
IMS	Atur bendera interrupt	IF = 1;	0xFB
STOSB	Simpan byte dalam string	`if ( DF == 0 ) * ES : DI ++ = AL ; else * ES : DI -- = AL ;`	0xAA
STOSW	Simpan kata dalam string	`if ( DF == 0 ) * ES : DI ++ = AX ; else * ES : DI -- = AX ;`	0xAB
SUB	Pengurangan	(1) r/m -= r/imm; (2) r -= m/imm;	0x28 ... 0x2D, 0x80 ... 0x83 / 5
UJI	Perbandingan logis (DAN)	(1) r/m & r/imm; (2) r & m/imm;	0x84, 0x84, 0xA8, 0xA9, 0xF6 / 0, 0xF7 / 0

TUNGGU	Tunggu sampai tidak sibuk	Menunggu hingga pin BUSY # tidak aktif (digunakan dengan unit floating-point)	0x9B
XCHG	Tukar data	r : = : r / m ; Spinlock biasanya menggunakan xchg sebagai operasi atom . (bug koma).	0x86, 0x87, 0x91 ... 0x97
XLAT	Terjemahan pencarian tabel	berperilaku seperti MOV AL, [BX+AL]	0xD7
XOR	Eksklusif ATAU	(1) r/m ^ = r/imm; (2) r ^ = m/imm;	0x30 ... 0x35, 0x80 ... 0x83 / 6