

LAPORAN TUGAS PRAKTIKUM MODUL 1
PRAKTIKUM SISTEM OPERASI



Nama : Indah Setia Rini

NIM : L200190265

Kelas : G

UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN PERLAJARAN 2020

Tugas

1. Apa yang dimaksud kode 'ASCII', buatlah table kode ASCII lengkap cukup dengan kode ASCII yang standar tidak perlu extend, tuliskan kode ASCII dalam format angka decimal, binary, dan hexadecimal serta karakter dan symbol yang dikodekan. **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.

Nilai ANSI ASCII (Desimal)	Nilai Unicode (Heksa Desimal)	Karakter	Keterangan
0	0000	NUL	Null (tidak terlihat)
1	0001	SOH	Start of heading (tidak terlihat)
2	0002	STX	Start of text (tidak terlihat)
3	0003	ETX	End of text (tidak terlihat)
4	0004	EOT	End of transmission (tidak terlihat)
5	0005	ENQ	Enquiry (tidak terlihat)
6	0006	ACK	Acknowledge (tidak terlihat)
7	0007	BEL	Bell (tidak terlihat)
8	0008	BS	Backspace
9	0009	HT	Horizontal tabulation
10	000A	LF	Pergantian baris (Line feed)
11	000B	VT	Tabulasi vertikal
12	000C	FF	Pergantian baris (Form feed)
13	000D	CR	Pergantian baris (carriage return)
14	000E	SO	Shift out (tidak terlihat)
15	000F	SI	Shift in (tidak terlihat)
16	0010	DLE	Data link escape (tidak terlihat)
17	0011	DC1	Device control 1 (tidak terlihat)
18	0012	DC2	Device control 2 (tidak terlihat)
19	0013	DC3	Device control 3 (tidak terlihat)
20	0014	DC4	Device control 4 (tidak terlihat)

21	0015	NAK	Negative acknowledge (tidak terlihat)
22	0016	SYN	Synchronous idle (tidak terlihat)
23	0017	ETB	End of transmission block (tidak terlihat)
24	0018	CAN	Cancel (tidak terlihat)
25	0019	EM	End of medium (tidak terlihat)
26	001A	SUB	Substitute (tidak terlihat)
27	001B	ESC	Escape (tidak terlihat)
28	001C	FS	File separator
29	001D	GS	Group separator
30	001E	RS	Record separator
31	001F	US	Unit separator
32	0020	spasi	Spasi
33	0021	!	Tanda seru (exclamation)
34	0022	“	Tanda kuti dua
35	0023	#	Tanda pagar (kres)
36	0024	\$	Tanda mata uang dolar
37	0025	%	Tanda persen
38	0026	&	Karakter ampersand (&)
39	0027	'	Karakter Apostrof
40	0028	(Tanda kurung buka
41	0029)	Tanda kurung tutup
42	002A	*	Karakter asterisk (bintang)

43	002B	+	Tanda tambah (plus)
44	002C	,	Karakter koma
45	002D	-	Karakter hyphen (strip)
46	002E	.	Tanda titik
47	002F	/	Garis miring (slash)
48	0030	0	Angka nol
49	0031	1	Angka satu
50	0032	2	Angka dua
51	0033	3	Angka tiga
52	0034	4	Angka empat
53	0035	5	Angka lima
54	0036	6	Angka enam
55	0037	7	Angka tujuh
56	0038	8	Angka delapan
57	0039	9	Angka sembilan
58	003A	:	Tanda titik dua
59	003B	;	Tanda titik koma
60	003C	<	Tanda lebih kecil
61	003D	=	Tanda sama dengan
62	003E	>	Tanda lebih besar
63	003F	?	Tanda tanya
64	0040	@	A keong (@)
65	0041	A	Huruf latin A kapital

66	0042	B	Huruf latin B kapital
67	0043	C	Huruf latin C kapital
68	0044	D	Huruf latin D kapital
69	0045	E	Huruf latin E kapital
70	0046	F	Huruf latin F kapital
71	0047	G	Huruf latin G kapital
72	0048	H	Huruf latin H kapital
73	0049	I	Huruf latin I kapital
74	004A	J	Huruf latin J kapital
75	004B	K	Huruf latin K kapital
76	004C	L	Huruf latin L kapital
77	004D	M	Huruf latin M kapital
78	004E	N	Huruf latin N kapital
79	004F	O	Huruf latin O kapital
80	0050	P	Huruf latin P kapital
81	0051	Q	Huruf latin Q kapital
82	0052	R	Huruf latin R kapital
83	0053	S	Huruf latin S kapital
84	0054	T	Huruf latin T kapital
85	0055	U	Huruf latin U kapital
86	0056	V	Huruf latin V kapital
87	0057	W	Huruf latin W kapital
88	0058	X	Huruf latin X kapital

89	0059	Y	Huruf latin Y kapital
90	005A	Z	Huruf latin Z kapital
91	005B	[Kurung siku kiri
92	005C	/	Garis miring terbalik (<i>backslash</i>)
93	005D]	Kurung sikur kanan
94	005E	^	Tanda pangkat
95	005F	_	Garis bawah (<i>underscore</i>)
96	0060	`	Tanda petik satu
97	0061	a	Huruf latin a kecil
98	0062	b	Huruf latin b kecil
99	0063	c	Huruf latin c kecil
100	0064	d	Huruf latin d kecil
101	0065	e	Huruf latin e kecil
102	0066	f	Huruf latin f kecil
103	0067	g	Huruf latin g kecil
104	0068	h	Huruf latin h kecil
105	0069	i	Huruf latin i kecil
106	006A	j	Huruf latin j kecil
107	006B	k	Huruf latin k kecil
108	006C	l	Huruf latin l kecil
109	006D	m	Huruf latin m kecil
110	006E	n	Huruf latin n kecil
111	006F	o	Huruf latin o kecil

112	0070	p	Huruf latin p kecil
113	0071	q	Huruf latin q kecil
114	0072	r	Huruf latin r kecil
115	0073	s	Huruf latin s kecil
116	0074	t	Huruf latin t kecil
117	0075	u	Huruf latin u kecil
118	0076	v	Huruf latin v kecil
119	0077	w	Huruf latin w kecil
120	0078	x	Huruf latin x kecil
121	0079	y	Huruf latin y kecil
122	007A	z	Huruf latin z kecil
123	007B	{	Kurung kurawal buka
124	007C		Garis vertikal (pipa)
125	007D	}	Kurung kurawal tutup
126	007E	~	Karakter gelombang (tilde)
127	007F	DEL	Delete
128	0080	€	Euro sign
129	0081		
130	0082	,	Single low-9 quotation mark
131	0083	f	Latin small letter f with hook
132	0084	„	Double low-9 quotation mark
133	0085	...	Horizontal ellipsis
134	0086	†	Dagger
135	0087	‡	Double dagger

136	0088	ˆ	Modifier letter circumflex accent
137	0089	‰	Per mille sign
138	008A	Š	Latin capital letter S with caron
139	008B	◄	Single left-pointing angle quotation
140	008C	Œ	Latin capital ligature OE
141	008D		
142	008E	Ž	Latin capital letter Z with caron
143	008F		
144	0090		
145	0091	‘	Left single quotation mark
146	0092	’	Right single quotation mark
147	0093	“	Left double quotation mark
148	0094	”	Right double quotation mark
149	0095	•	Bullet
150	0096	—	En dash
151	0097	—	Em dash
152	0098	˜	Small tilde
153	0099	™	Trade mark sign
154	009A	š	Latin small letter S with caron
155	009B	›	Single right-pointing angle quotation mark
156	009C	œ	Latin small ligature oe
157	009D		
158	009E	ž	Latin small letter z with caron
159	009F	ÿ	Latin capital letter Y with diaeresis

2. Carilah daftar perintah Bahasa assembly untuk mesin intel keluarga x86 lengkap, (dari buku referensi atau internet). Daftar perintah ini dapat digunakan sebagai pedoman untuk membantu memahami program ‘boot.asm’ dan ‘kernel.asm’.

DB : define bytes. Membentuk data byte demi byte. Data bisa data numerik maupun teks.

DD : define double words. Membentuk data doubleword demi doubleword (4 byte).

MOV : move. Memindahkan suatu nilai dari register ke memori, memori ke register, atau register ke register.

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

AND : melakukan bitwise and. sintaks: AND {register}, {angka} AND {register 1}, {register 2} hasil disimpan di register 1.

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.

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 : subtract. Mengurangkan dua buah register.

sintaks: SUB {tujuan}, {sumber} operasi yang terjadi: tujuan = tujuan – sumber.
borrow (bila terjadi) menyebabkan CF bernilai 1.

SBB : subtract 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 $CX \neq 0$ dan $ZF = 1$. CX tetap dikurangi 1 sebelum diperiksa.

sintaks: LOOPE {label tujuan}

LOOPZ : loop while zero. Identik dengan LOOPE.

LOOPNE : loop while not equal.

Melakukan pengulangan selama $CX \neq 0$ 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.

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 membackup 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.

PUSHA : push all general-purpose registers. Lawan dari POPA,

dimana PUSHAX 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

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 program 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 dengan berbagai cara.

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.

sintaks: CALL {label nama prosedur}

RET : return. Tanda selesai prosedur.

Setiap prosedur harus memiliki RET di ujungnya.

sintaks: RET ;(saja)

HLT : halt. Membuat prosesor menjadi tidak aktif.