TUGAS PRAKTIKUM MODUL 1
PENGENALAN SISTEM PENGEMBANGAN OS DENGAN PC SIMULATOR 'BOCHS'

Dosen Pengampu
Heru Setya Nugraha, S.T., M.Kom.

Mata Kuliah
Praktikum Sistem Operasi

Disusun oleh
Said Mohammad Ghuzi
L200190126

TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN AJARAN 2020-2021

TUGAS 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.
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 memahami program ‘boot.asm’ dan ‘kernel.asm’.

JAWABAN

1. ASCII (American Standard Code for Information Interchange) atau Kode Standar Amerika untuk Pertukaran Informasi adalah standar pengkodean karakter untuk alat komunikasi. Kode ASCII mewakili teks dalam komputer, peralatan telekomunikasi, dan perangkat lainnya. Kebanyakan skema pengkodean karakter modern didasarkan pada ASCII, meskipun mereka mendukung banyak karakter tambahan.

TABLE KODE ASCII

Desimal	Karakter	Heksa Desimal	Biner
0	NUL	0000	0000 0000
1	SOH	0001	0000 0001
2	STX	0002	0000 0010
3	ETX	0003	0000 0011
4	EOT	0004	0000 0100
5	ENQ	0005	0000 0101
6	ACK	0006	0000 0110
7	BEL	0007	0000 0111
8	BS	0008	0000 1000
9	HT	0009	0000 1001
10	LF	000A	0000 1010
11	VT	000B	0000 1011
12	FF	000C	0000 1100
13	CR	000D	0000 1101
14	SO	000E	0000 1110
15	SI	000F	0000 1111
16	DLE	0010	0001 0000
17	DC1	0011	0001 0001
18	DC2	0012	0001 0010
19	DC3	0013	0001 0011
20	DC4	0014	0001 0100
21	NAK	0015	0001 0101
22	SYN	0016	0001 0110

23	ETB	0017	0001 0111
24	CAN	0018	0001 1000
25	EM	0019	0001 1001
26	SUB	001A	0001 1010
27	ESC	001B	0001 1011
28	FS	001C	0001 1100
29	GS	001D	0001 1101
30	RS	001E	0001 1110
31	US	001F	0001 1111
32	spasi	0020	0010 0000
33	!	0021	0010 0001
34	“	0022	0010 0010
35	#	0023	0010 0011
36	$	0024	0010 0100
37	%	0025	0010 0101
38	&	0026	0010 0110
39	‘	0027	0010 0111
40	(	0028	0010 1000
41	)	0029	0010 1001
42	*	002A	0010 1010
43	+	002B	0010 1011
44	,	002C	0010 1100
45	-	002D	0010 1101
46	.	002E	0010 1110
47	/	002F	0010 1111
48	0	0030	0011 0000
49	1	0031	0011 0001
50	2	0032	0011 0010
51	3	0033	0011 0011
52	4	0034	0011 0100
53	5	0035	0011 0101
54	6	0036	0011 0110
55	7	0037	0011 0111
56	8	0038	0011 1000
57	9	0039	0011 1001
58	:	003A	0011 1010
59	;	003B	0011 1011
60	<	003C	0011 1100
61	=	003D	0011 1101
62	>	003E	0011 1110
63	?	003F	0011 1111
64	@	0040	0100 0000
65	A	0041	0100 0001
66	B	0042	0100 0010
67	C	0043	0100 0011

68	D	0044	0100 0100
69	E	0045	0100 0101
70	F	0046	0100 0110
71	G	0047	0100 0111
72	H	0048	0100 1000
73	I	0049	0100 1001
74	J	004A	0100 1010
75	K	004B	0100 1011
76	L	004C	0100 1100
77	M	004D	0100 1101
78	N	004E	0100 1110
79	O	004F	0100 1111
80	P	0050	0101 0000
81	Q	0051	0101 0001
82	R	0052	0101 0010
83	S	0053	0101 0011
84	T	0054	0101 0100
85	U	0055	0101 0101
86	V	0056	0101 0110
87	W	0057	0101 0111
88	X	0058	0101 1000
89	Y	0059	0101 1001
90	Z	005A	0101 1010
91	[	005B	0101 1011
92	/	005C	0101 1100
93	]	005D	0101 1101
94	^	005E	0101 1110
95	_	005F	0101 1111
96	`	0060	0110 0000
97	a	0061	0110 0001
98	b	0062	0110 0010
99	c	0063	0110 0011
100	d	0064	0110 0100
101	e	0065	0110 0101
102	f	0066	0110 0110
103	g	0067	0110 0111
104	h	0068	0110 1000
105	i	0069	0110 1001
106	j	006A	0110 1010
107	k	006B	0110 1011
108	l	006C	0110 1100
109	m	006D	0110 1101
110	n	006E	0110 1110
111	o	006F	0110 1111
112	p	0070	0111 0000

113	q	0071	0111 0001
114	r	0072	0111 0010
115	s	0073	0111 0011
116	t	0074	0111 0100
117	u	0075	0111 0101
118	v	0076	0111 0110
119	w	0077	0111 0111
120	x	0078	0111 1000
121	y	0079	0111 1001
122	z	007A	0111 1010
123	{	007B	0111 1011
124	|	007C	0111 1100
125	}	007D	0111 1101
126	~	007E	0111 1110
127	DEL	007F	0111 1111

2. **Bahasa Assembly Intel Keluarga x86**

- ACALL (Absolute Call)
- ADD (Add Immediate Data)
- ADDC (Add Carry Plus Immediate Data to Accumulator)
- AJMP (Absolute Jump)
- ANL (Logical AND memori ke akumulator)
- CJNE (Compare Indirect Address to Immediate Data)
- CLR (Clear Accumulator)
- CPL (Complement Accumulator)
- DA (Decimal Adjust Accumulator)
- DEC (Decrement Indirect Address)
- DIV (Divide Accumulator by B)
- DJNZ (Decrement Register And Jump Id Not Zero)
- INC (Increment Indirect Address)
- JB (Jump if Bit is Set)
- JBC (Jump if Bit Set and Clear Bit)
- JC (Jump if Carry is Set)
- JMP (Jump to sum of Accumulator and Data Pointer)
- JNB (Jump if Bit is Not Set)
- JNC (Jump if Carry Not Set)
- JNZ (Jump if Accumulator Not Zero)
- JZ (Jump if Accumulator is Zero)
- LCALL (Long Call)
- LJMP (Long Jump)

- MOV (Move From Memory)
- MOVC (Move From Codec Memory)
- MOVX (Move Accumulator to External Memory Addressed by Data Pointer)
- MUL (Multiply)
- NOP (No Operation)
- ORL (Logical OR Immediate Data to Accumulator)
- POP (Pop Stack to Memory)
- PUSH (Push Memory onto Stack)
- RET (Return from subroutine)
- RETI (Return From Interrupt)
- RL (Rotate Accumulator Left)
- RLC (Rotate Left through Carry)
- RR (Rotate Right)
- RRC (Rotate Right through Carry)
- SETB (set Carry flag)
- SJMP (Short Jump)
- SUBB (Subtract With Borrow)
- SWAP (Swap Nibbles)
- XCH (Exchange Bytes)
- XCHD (Exchange Digits)
- XRL (Exclusive OR Logic)