UNIVERSITI MALAYSIA PERLIS

Peperiksaan Pertengahan Semester Pertama
Sidang Akademik 2019/2020

DKT 218 – Microcontroller
[Mikropengawal]

Masa: 2 jam

Please make sure that this question paper has **NINE (9)** printing pages including this front page before you start the examination.
(*Sila pastikan kertas soalan ini mengandungi **SEMBILAN (9)** mukasurat yang bercetak termasuk muka hadapan sebelum anda memulakan peperiksaan ini.*).

This question paper has **FOUR (4)** questions. Answer **ALL** questions.
(*Kertas soalan ini mengandungi **EMPAT (4)** soalan. Jawab **SEMUA** soalan.*

QUESTION 1

Figure 1 shows an 8051 microcontroller package chip developed by Intel that has a combined features which differs from a standard microprocessor.

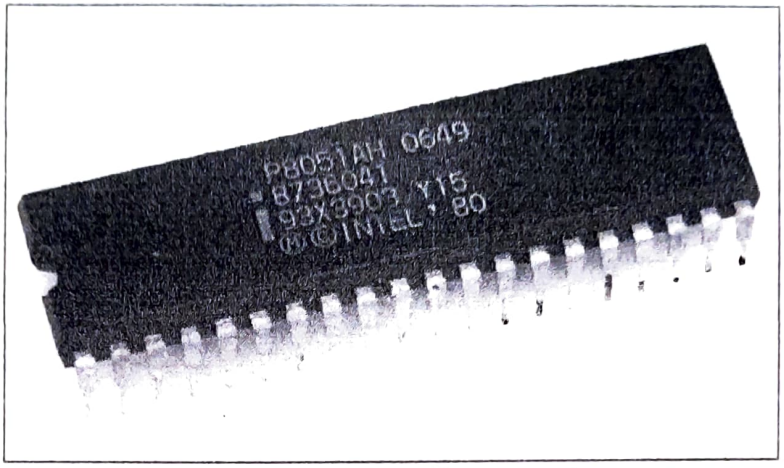

Figure 1 : Intel 8051 microcontroller

(a) Draw a block diagram of the internal features for this 8051 microcontroller.

[4 Marks]

(b) List down six of these feature's charactertistics as provided in Table 1a.

Table 1a

No	Feature	Quantity / Size
1	P. Memory	4KB
2	Data memory	128 bytes
3	Data bus	8-bit
4	Addr Bus	16-bit
5	External interrupt	2
6	Internal	2

[6 Marks]

QUESTION 2

a) Table 1 below shows an 8051 assembly program for a certain mathematical procedure. Convert the assembly program codes into machine codes.

Table 1

Line	Assembly Program		
1		ORG	100H
2		MOV	SP,#74H
3	AGAIN:	MOV	PSW,#00H
4		SETB	PSW.3
5	LOOP1:	CLR	C
6		MOV	A,24H
7		SUBB	A,34H
8		JZ	LOOP1
9		JC	LOOP2
10		MOV	R7,24H
11		SJMP	AGAIN
12	LOOP2:	MOV	R7,34H
13		SJMP	AGAIN
14		END	

[5 Marks]

b) Based on the 8051 assembly program in Table 2.1, show the calculation to determine the offset value for the machine codes at lines 8, 9 and 13.

[3 Marks]

c) Determine the register values (in hex) for PSW, Accumulator and R7 if the contents of RAM addresses 24H and 34H are 33H and 37H respectively.

[2 Marks]

QUESTION 3

Figure 2 below shows an 8051 microcontroller assembly language delay program using the single-nested loop.

```
DELAY:  NOP
        MOV     R3,#200
        NOP
LOOP1:  NOP
        NOP
        NOP
        DJNZ    R3,LOOP1
        NOP
        MOV     R3,#100
LOOP2:  NOP
        DJNZ    R3,LOOP2
        NOP
        RET
```

Figure 2 : Delay program

a) Determine the total number of machine cycles for the entire program.
[3 Marks]

b) If the system uses a 24MHz crystal clock source, determine the delay's time length.
[2 Marks]

c) Write a program to blink an LED at Port 0.3 using the delay program above.
[5 Marks]

QUESTION 4

A 0.5Hz frequency alarm buzzer is to be connected to P1.2 of the 8051 microcontroller, Atmel AT89S52, and supplied with a 120kHz crystal oscillator, X1.

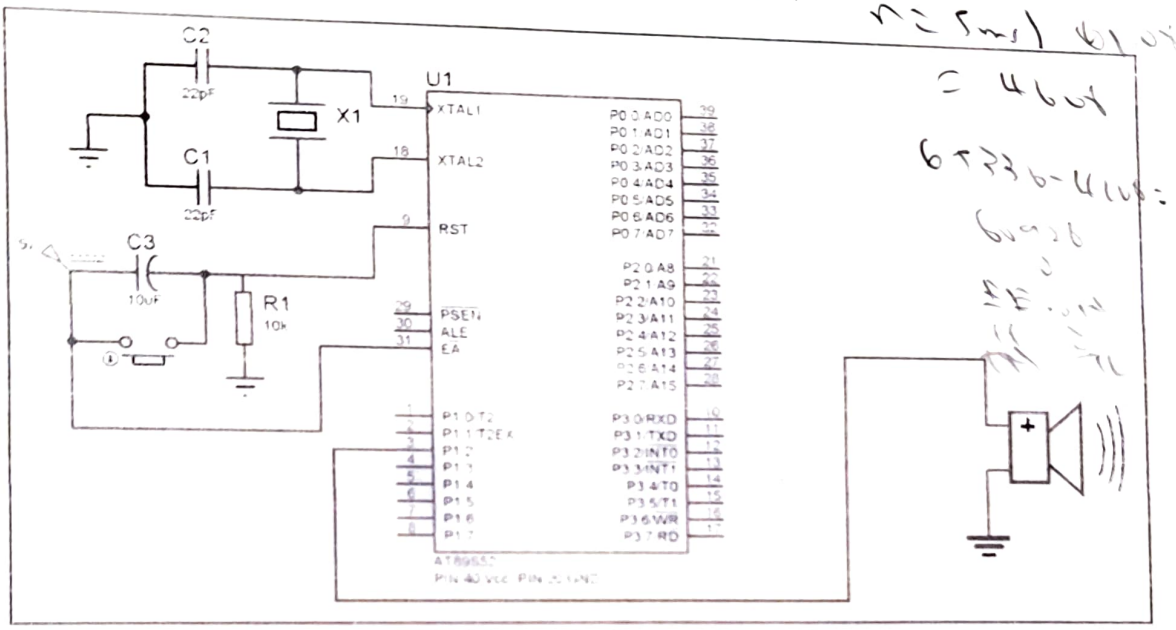

Figure 4a : 8051 Schematic Diagram Buzzer System

a) Using the Counter/Timer MOD 1 of Timer 0 features,

i. Determine the microcontroller's period machine cycle **[2 Marks]**

ii. Determine the hexadecimal value to be stored in TH0 and TL0. **[3 Marks]**

iii. Create the assembly program to initialize and setup the proper Timer as required. **[3 Marks]**

iv. Create the assembly program that will generate the proper frequency source of the alarm. **[6 Marks]**

b) If the Interrupt feature is used for this system, create the program using the same Timer features in (a). **[6 Marks]**

APPENDIX 1

HEX CODES

HEX Code	Mnemonic	Operand	Byte	Cycle	C	OV	AC
00	NOP		1	1			
01	AJMP	addr11	2	2			
02	LJMP	addr16	3	2			
03	RR	A	1	1			
04	INC	A	1	1			
05	INC	direct	2	1			
06	INC	@R0	1	1			
07	INC	@R1	1	1			
08	INC	R0	1	1			
09	INC	R1	1	1			
0A	INC	R2	1	1			
0B	INC	R3	1	1			
0C	INC	R4	1	1			
0D	INC	R5	1	1			
0E	INC	R6	1	1			
0F	INC	R7	1	1			
10	JBC	bit, offset	3	2			
11	ACALL	addr11	2	2			
12	LCALL	addr16	3	2			
13	RRC	A	1	1	x		
14	DEC	A	1	1			
15	DEC	direct	2	1			
16	DEC	@R0	1	1			
17	DEC	@R1	1	1			
18	DEC	R0	1	1			
19	DEC	R1	1	1			
1A	DEC	R2	1	1			
1B	DEC	R3	1	1			
1C	DEC	R4	1	1			
1D	DEC	R5	1	1			
1E	DEC	R6	1	1			
1F	DEC	R7	1	1			
20	JB	bit, offset	3	2			
21	AJMP	addr11	2	2			
22	RET		1	2			
23	RL	A	1	1			
24	ADD	A, #immed	2	1	x	x	x
25	ADD	A, direct	2	1	x	x	x
26	ADD	A, @R0	1	1	x	x	x
27	ADD	A, @R1	1	1	x	x	x
28	ADD	A, R0	1	1	x	x	x
29	ADD	A, R1	1	1	x	x	x
2A	ADD	A, R2	1	1	x	x	x
2B	ADD	A, R3	1	1	x	x	x
2C	ADD	A, R4	1	1	x	x	x
2D	ADD	A, R5	1	1	x	x	x
2E	ADD	A, R6	1	1	x	x	x
2F	ADD	A, R7	1	1	x	x	x
30	JNB	bit, offset	3	2			
31	ACALL	addr11	2	2			
32	RETI		1	2			
33	RLC	A	1	1	x		
34	ADDC	A, #immed	2	1	x	x	x
35	ADDC	A, direct	2	1	x	x	x
36	ADDC	A, @R0	1	1	x	x	x
37	ADDC	A, @R1	1	1	x	x	x
38	ADDC	A, R0	1	1	x	x	x
39	ADDC	A, R1	1	1	x	x	x
3A	ADDC	A, R2	1	1	x	x	x
3B	ADDC	A, R3	1	1	x	x	x
3C	ADDC	A, R4	1	1	x	x	x
3D	ADDC	A, R5	1	1	x	x	x
3E	ADDC	A, R6	1	1	x	x	x
3F	ADDC	A, R7	1	1	x	x	x

HEX Code	Mnemonic	Operand	Byte	Cycle	C	OV	AC
40	JC	offset	2	2			
41	AJMP	addr11	2	2			
42	ORL	direct, A	2	1			
43	ORL	direct, #immed	3	2			
44	ORL	A, #immed	2	1			
45	ORL	A, direct	2	1			
46	ORL	A, @R0	1	1			
47	ORL	A, @R1	1	1			
48	ORL	A, R0	1	1			
49	ORL	A, R1	1	1			
4A	ORL	A, R2	1	1			
4B	ORL	A, R3	1	1			
4C	ORL	A, R4	1	1			
4D	ORL	A, R5	1	1			
4E	ORL	A, R6	1	1			
4F	ORL	A, R7	1	1			
50	JNC	offset	2	2			
51	ACALL	addr11	2	2			
52	ANL	direct, A	2	1			
53	ANL	direct, #immed	3	2			
54	ANL	A, #immed	2	1			
55	ANL	A, direct	2	1			
56	ANL	A, @R0	1	1			
57	ANL	A, @R1	1	1			
58	ANL	A, R0	1	1			
59	ANL	A, R1	1	1			
5A	ANL	A, R2	1	1			
5B	ANL	A, R3	1	1			
5C	ANL	A, R4	1	1			
5D	ANL	A, R5	1	1			
5E	ANL	A, R6	1	1			
5F	ANL	A, R7	1	1			
60	JZ	offset	2	2			
61	AJMP	addr11	2	2			
62	XRL	direct, A	2	1			
63	XRL	direct, #immed	3	2			
64	XRL	A, #immed	2	1			
65	XRL	A, direct	2	1			
66	XRL	A, @R0	1	1			
67	XRL	A, @R1	1	1			
68	XRL	A, R0	1	1			
69	XRL	A, R1	1	1			
6A	XRL	A, R2	1	1			
6B	XRL	A, R3	1	1			
6C	XRL	A, R4	1	1			
6D	XRL	A, R5	1	1			
6E	XRL	A, R6	1	1			
6F	XRL	A, R7	1	1			
70	JNZ	offset	2	2			
71	ACALL	addr11	2	2			
72	ORL	C, bit	2	2	x		
73	JMP	@A+DPTR	1	2			
74	MOV	A, #immed	2	1			
75	MOV	direct, #immed	3	2			
76	MOV	@R0, #immed	2	1			
77	MOV	@R1, #immed	2	1			
78	MOV	R0, #immed	2	1			
79	MOV	R1, #immed	2	1			
7A	MOV	R2, #immed	2	1			
7B	MOV	R3, #immed	2	1			
7C	MOV	R4, #immed	2	1			
7D	MOV	R5, #immed	2	1			
7E	MOV	R6, #immed	2	1			
7F	MOV	R7, #immed	2	1			

HEX Code	Mnemonic	Operand	Byte	Cycle	C	OV	AC
80	SJMP	offset	2	2			
81	AJMP	addr11	2	2			
82	ANL	C, bit	2	2	x		
83	MOVC	A, @A+PC	1	2			
84	DIV	AB	1	4	0	x	
85	MOV	direct, direct	3	2			
86	MOV	direct, @R0	2	2			
87	MOV	direct, @R1	2	2			
88	MOV	direct, R0	2	2			
89	MOV	direct, R1	2	2			
8A	MOV	direct, R2	2	2			
8B	MOV	direct, R3	2	2			
8C	MOV	direct, R4	2	2			
8D	MOV	direct, R5	2	2			
8E	MOV	direct, R6	2	2			
8F	MOV	direct, R7	2	2			
90	MOV	DPTR, #immed	3	2			
91	ACALL	addr11	2	2			
92	MOV	bit, C	2	2			
93	MOVC	A, @A+DPTR	1	2			
94	SUBB	A, #immed	2	1	x	x	x
95	SUBB	A, direct	2	1	x	x	x
96	SUBB	A, @R0	1	1	x	x	x
97	SUBB	A, @R1	1	1	x	x	x
98	SUBB	A, R0	1	1	x	x	x
99	SUBB	A, R1	1	1	x	x	x
9A	SUBB	A, R2	1	1	x	x	x
9B	SUBB	A, R3	1	1	x	x	x
9C	SUBB	A, R4	1	1	x	x	x
9D	SUBB	A, R5	1	1	x	x	x
9E	SUBB	A, R6	1	1	x	x	x
9F	SUBB	A, R7	1	1	x	x	x
A0	ORL	C, /bit	2	2	x		
A1	AJMP	addr11	2	2			
A2	MOV	C, bit	2	1	X		
A3	INC	DPTR	1	2			
A4	MUL	AB	1	4	0	x	
A5	reserved						
A6	MOV	@R0, direct	2	2			
A7	MOV	@R1, direct	2	2			
A8	MOV	R0, direct	2	2			
A9	MOV	R1, direct	2	2			
AA	MOV	R2, direct	2	2			
AB	MOV	R3, direct	2	2			
AC	MOV	R4, direct	2	2			
AD	MOV	R5, direct	2	2			
AE	MOV	R6, direct	2	2			
AF	MOV	R7, direct	2	2			
B0	ANL	C, /bit	2	2	X		
B1	ACALL	addr11	2	2			
B2	CPL	bit	2	1			
B3	CPL	C	1	1	X		
B4	CJNE	A, #immed, offset	3	2	x		
B5	CJNE	A, direct, offset	3	2	x		
B6	CJNE	@R0, #immed, offset	3	2	x		
B7	CJNE	@R1, #immed, offset	3	2	x		
B8	CJNE	R0, #immed, offset	3	2	x		
B9	CJNE	R1, #immed, offset	3	2	x		
BA	CJNE	R2, #immed, offset	3	2	x		
BB	CJNE	R3, #immed, offset	3	2	x		
BC	CJNE	R4, #immed, offset	3	2	x		
BD	CJNE	R5, #immed, offset	3	2	x		
BE	CJNE	R6, #immed, offset	3	2	x		
BF	CJNE	R7, #immed, offset	3	2			

HEX Code	Mnemonic	Operand	Byte	Cycle	C	OV	AC
C0	PUSH	direct	2	2			
C1	AJMP	addr11	2	2			
C2	CLR	bit	2	1			
C3	CLR	C	1	1	0		
C4	SWAP	A	1	1			
C5	XCH	A, direct	2	1			
C6	XCH	A, @R0	1	1			
C7	XCH	A, @R1	1	1			
C8	XCH	A, R0	1	1			
C9	XCH	A, R1	1	1			
CA	XCH	A, R2	1	1			
CB	XCH	A, R3	1	1			
CC	XCH	A, R4	1	1			
CD	XCH	A, R5	1	1			
CE	XCH	A, R6	1	1			
CF	XCH	A, R7	1	1			
D0	POP	direct	2	2			
D1	ACALL	addr11	2	2			
D2	SETB	bit	2	1			
D3	SETB	C	1	1	1		
D4	DA	A	1	1	x		
D5	DJNZ	direct, offset	3	2			
D6	XCHD	A, @R0	1	1			
D7	XCHD	A, @R1	1	1			
D8	DJNZ	R0, offset	2	2			
D9	DJNZ	R1, offset	2	2			
DA	DJNZ	R2, offset	2	2			
DB	DJNZ	R3, offset	2	2			
DC	DJNZ	R4, offset	2	2			
DD	DJNZ	R5, offset	2	2			
DE	DJNZ	R6, offset	2	2			
DF	DJNZ	R7, offset	2	2			
E0	MOVX	A, @DPTR	1	2			
E1	AJMP	addr11	2	2			
E2	MOVX	A, @R0	1	2			
E3	MOVX	A, @R1	1	2			
E4	CLR	A	1	1			
E5	MOV	A, direct	2	1			
E6	MOV	A, @R0	1	1			
E7	MOV	A, @R1	1	1			
E8	MOV	A, R0	1	1			
E9	MOV	A, R1	1	1			
EA	MOV	A, R2	1	1			
EB	MOV	A, R3	1	1			
EC	MOV	A, R4	1	1			
ED	MOV	A, R5	1	1			
EE	MOV	A, R6	1	1			
EF	MOV	A, R7	1	1			
F0	MOVX	@DPTR, A	1	2			
F1	ACALL	addr11	2	2			
F2	MOVX	@R0, A	1	2			
F3	MOVX	@R1, A	1	2			
F4	CPL	A	1	1			
F5	MOV	direct, A	2	1			
F6	MOV	@R0, A	1	1			
F7	MOV	@R1, A	1	1			
F8	MOV	R0, A	1	1			
F9	MOV	R1, A	1	1			
FA	MOV	R2, A	1	1			
FB	MOV	R3, A	1	1			
FC	MOV	R4, A	1	1			
FD	MOV	R5, A	1	1			
FE	MOV	R6, A	1	1			
FF	MOV	R7, A	1	1			

APPENDIX 2

8051 SPECIAL FUNCTION REGISTERS

Reg Type									Byte address
SCON	9F	9E	9D	9C	9B	9A	99	98	98
P1	97	96	95	94	93	92	91	90	90
TH1	not bit addressable								8D
TH0	not bit addressable								8C
TL1	not bit addressable								8B
TL0	not bit addressable								8A
TMOD	not bit addressable								89
TCON	8F	8E	8D	8C	8B	8A	89	88	88
PCON	not bit addressable								87
DPH	not bit addressable								83
DPL	not bit addressable								82
SP	not bit addressable								81
P0	87	86	85	84	83	82	81	80	80

Reg Type									Byte address
									FF
B	F7	F6	F5	F4	F3	F2	F1	F0	F0
ACC	E7	E6	E5	E4	E3	E2	E1	E0	E0
PSW	D7	D6	D5	D4	D3	D2	--	D0	D0
IP	--	--	--	BC	BB	BA	B9	B8	B8
P3	B7	B6	B5	B4	B3	B2	B1	B0	B0
IE	AF	--	--	AC	AB	AA	A9	A8	A8
P2	A7	A6	A5	A4	A3	A2	A1	A0	A0
SBUF	not bit addressable								99

SPECIAL FUNCTION REGISTER - PSW

BYTE	BIT							
D0	D7	D6	D5	D4	D3	D2	D1	D0
	PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
	CY	AC	F0	RS1	RS0	OV	--	P

SPECIAL FUNCTION REGISTER – TIMER

TCON REGISTER

BYTE	BIT							
88	8F	8E	8D	8C	8B	8A	89	88
	TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

IT_X – external interrupt type ; $edge/\overline{level}$ triggered
IE_X – external flag; HIGH when detect falling-edge until RETI
TR_X – timer run ; $start/\overline{stop}$ bit
TF_X – timer flag; HIGH everytime rollover to 00H

TMOD REGISTER

BYTE	BIT							
89	--	--	--	--	--	--	--	--
	TMOD							
	G	$C/\bar{T}$	M1	M0	G	$C/\bar{T}$	M1	M0

G – gate control; '1' – timer runs when IE_X='1' and TR_X='1'
'0' – timer runs when TR_X='1' only
$C/\bar{T}$ – whether to use $\overline{internal}/external$ clock soure
M1 M0 – modes "00"-13 bit, "01"-16 bit, "10"-8 bit, "11"-split timer

SPECIAL FUNCTION REGISTER – INTERRUPT

IE REGISTER

BYTE	BIT							
A8	AF	AE	AD	AC	AB	AA	A9	A8
	IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
	EA	--	--	ES	ET1	EX1	ET0	EX0

IVT

Type	Reset	Ex0	T0	Ex1	T1	S
Address	0000	0003	000B	0013	001B	0023

IP REGISTER

BYTE	BIT							
B8	BF	BE	BD	BC	BB	BA	B9	B8
	IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
	--	--	--	PS	PT1	PX1	PT0	PX0

SPECIAL FUNCTION REGISTER – SERIAL COMMUNICATION

SCON REGISTER

BYTE	BIT							
98	9F	9E	9D	9C	9B	9A	99	98
	SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

RI – Sets when a byte has been received in SBUFF
TI – Sets when a byte has been transmitted
RB8 – Receive 9th bit for mode 2 and 3.
TB8 – Transmit 9th bit for mode 2 and 3
REN – Receiver enable; HIGH to receive data on RxD pin
SM2 – Enables multiprocessor comm in mode 2 and 3
SM0 SM1 – "00" – Shift Reg baud rate OSC/12
"01" – 8-bit UART variable baud rate
"10" – 9-bit UART baud rate OSC/32 or OSC/64
"11" – 9-bit UART variable baud rate

PCON REGISTER

BYTE	BIT							
87	--	--	--	--	--	--	--	--
	PCON							
	SMOD	--	--	--	GF1	GF0	PD	IDL

SMOD – '0' -f/64 , '1' –f/32