EE309 – Microprocessors

Mid-Semester Exam, Autumn 2013 (IIT Bombay)

Wednesday, September 11, 2013; Duration: 11:00 – 13:00 Hrs; Maximum Marks: 25

1. Can we implement the funtionality of `RET` instruction in 8051 using some other instructions (i.e. without using `RET`). If no, why not? If yes, how (write the set of instructions)? *[2]*

2. Draw the gate/block level schematic to show the hardware for Timer/Counter 0 that can be derived from the TCON and TMOD configurations. *[1]*

3. Write a set of 8051 instructions for initialization of relavant registers/latches/flags/pins etc. to configure the Timer/Counter 0 to generate an interrupt when 1000 negative edges are detected on the T0 (P3.4) pin. You can refer the schematic of problem 2. *[2]*

4. For most 8085 instructions, the number of T-states required for the execution of an instruction is $3n + 1$, where n is the number of times the memory is read from or written to during fetch and execution of the instruction (provided the memory is NOT slow). However, the instruction `PUSH Rp` (a one-byte instruction that copies the contents of the register pair to the top of the stack) requires two T-states more than that suggested by this rule. *[1+1+1]*

   (a) How many T-states does the execution of this instruction require?

   (b) Why does the instruction require more number of cycles?

   (c) Given that the number of T-states required for `POP Rp` (which does the opposite of `PUSH Rp`) is given by the $3n + 1$ rule. Why does the same logic as mentioned for `PUSH Rp` not hold for `POP Rp`.

5. Assume for an 8085 microprocessor, a subroutine call instruction `CALL 4000H` is located in external memory with starting address 10FFH. Give the sequence of byte values available at the A15-A8 pins and the AD7-AD0 pins while this instruction is fetched and executed, if the value of SP just before the execution of this instruction is 1234H. *[4]*

6. You have to interface an 8051 microcontroller with an 8085 microprocessor such that the microcontroller acts like an IO device for the 8085 chip with IO port addresses 00H–7FH and 80H–FFH mapped to the 8051 RAM with addresses 00H–7FH (i.e. MSB of the port addresses can be ignored). For this implementation, the $\overline{\text{RD}}$, $\overline{\text{WR}}$, IO/$\overline{\text{M}}$ and READY signals from/to the 8085 chip, and the external interrupt $\overline{\text{INT0}}$ (P3.2) pin and some other port/pins of 8051 should be used for handshake etc. You can assume that the port pins of 8051 effectively provide high-impedance if ONEs are written to the corresponding port latches and the clock rates for both 8085 and 8051 are roughly same. *[2+4]*

   (a) Draw neatly the schematics showing a possible way to interface the two (you can assume that the external memory device used by 8085 is not slow and there is no need to show it in your schematics). You can also use some additional logic gates if required.

   (b) Write a set of initialization instructions and the $\overline{\text{INT0}}$ interrupt service subroutine for 8051 to implement the above functionality. The interrupt should be used in the edge triggered mode.

7. We wish to implement a counter in 8051 that counts from 0–9999 in decimal format. The upper two decimal digits are stored in the two nibbles of R1 register and the lower two decimal digits are stored in the two nibbles of R0 register (assume that flags RS1=0, RS0=0). Write a 8051 subroutine that increments this counter by one every time it is called. If the previous value was 9999, the counter should roll over to 0 and OV flag should be raised to 1. Also, assume that both R0 and R1 contain 0 initially. [Hint: Use DA instruction] *[3]*

8. Write a set of 8051 instructions to subtract the value 1234H from the contents of DPTR (the final result should be saved in DPTR only). If a borrow is required, the carry flag should be raised to 1 by the program. *[2]*

9. Assume that the P1.0 pin of an 8051 chip has been connected to ground externally using a wire. Consider the following instructions: *[1+1]*

```
SETB P1.0
CPL P1.0
MOV C, P1.0
```

   (a) What will be the state of the Carry flag after the above 3 instructions are executed and why?

   (b) What will be the state of Carry flag in case the wire to ground is removed after execution of the second instruction itself and why?

8051 instruction format: Instruction <dest>, <src>, <operand3>

Instruction	Possible Operands	Description	Oscillator Periods
Arithmetic Operations			
ADD	A,Rn A,direct A,@Ri A,#data	Add source to Accumulator	12
ADDC		Add source to Accumulator with Carry	12
SUBB		Subtract source from Accumulator with Borrow	12
INC	A Rn direct @Ri	Increment source	12
DEC		Decrement source	12
INC	DPTR	Increment 16-bit Data Pointer	24
MUL	AB	Multiply A & B (lower byte is stored in A)	48
DIV	AB	Divide A by B (Quotient is stored in A)	48
DA	A	Decimal Adjust Accumulator	12
Logical Operations			
ANL	A,Rn A,direct A,@Ri A,#data direct,A direct,#data	AND src to dest byte	24 if operands are "direct, #data"; 12 otherwise
ORL		OR src to dest byte	
XRL		Exclusive-OR src to dest byte	
CLR	A	Clear Accumulator	12
CPL		Complement Accumulator	12
RL		Rotate Accumulator Left	12
RLC		Rotate Accumulator Left through Carry	12
RR		Rotate Accumulator Right	12
RRC		Rotate Accumulaotr Right Through Carry	12
SWAP		Swap Nibbles within the Accumulator	12
Branching Instructions			
ACALL	addr11	Absolute subroutine call	24
LCALL	addr16	Long subroutine call	24
RET		Return from subroutine	24
RETI		Return from Interrupt	24
AJMP	addr11	Absolute jump	24
LJMP	addr16	Long jump	24
SJMP	rel	Short jump (relative addr)	24
JMP	@A+DPTR	Jump indirect relative to the DPTR	24
JZ	rel	Jump if Accumulator is Zero	24
JNZ	rel	Jump if Accumulator is Not Zero	24
CJNE	A,direct,rel A,#data,rel Rn,#data,rel @Ri,#data,rel	Compare the first two operands and jump of not equal; Set C if first operand < second operand, otherwise clear C	24
JNZ	Rn,rel	Decrement first operand and jump if not zero	24
	direct,rel		24
NOP		No Operation	12

Instruction	Possible Operands	Description	Oscillator Periods
Data Transfer			
MOV	A,Rn A,direct A,@Ri A,#data Rn,A Rn,#data direct,A @Ri,A @Ri,#data	Move src to dest	12
MOV	Rn,direct direct,Rn direct,direct direct,@Ri direct,#data @Ri,direct DPTR,#data16	Move src to dest	24
MOVC	A,@A+DPTR A,@A+PC	Move Code byte to Accumulator	24
MOVX	A,@Ri A,@DPTR @Ri,A @DPTR,A	Move External RAM byte to Accumulator	24
		Move Accumulator contents to External RAM	
PUSH	direct	Push direct byte to stack	24
POP		Pop direct byte from stack	
XCH	A,Rn A,direct A,@Ri	Exchange bytes	12
XCHD	A,@Ri	Exchange lower-order digits of the two operands	12
Boolean Variable Instructions			
CLR	C bit	Clear bit	12
SETB		Set bit	
CPL		Complement bit	
ANL	C,bit C,/bit	AND source to Carry	24
ORL		OR source bit to Carry	
RL		Rotate Accumulator Left	12
MOV	C,bit	Move src bit to dest bit	12
	bit,C		24
JC	rel	Jump if Carry is set	24
JNC		Jump if Carry is not set	
JB	bit,rel	Jump if bit is set	24
JNB		Jump if bit is NOT set	
JBC	bit,rel	Jump if Bit is set & clear it	24

Effects of some arithmetic instructions on flags

Instruction	Flag		
	C	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RRC	X		
RLC	X		

0 => Flag is cleared; X => Flag is affected

Special Function Registers

80	P0	SP	DPL	DPH				PCON	87
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
90	P1								97
98	SCON	SBUF							9F
A0	P2								A7
A8	IE								AF
B0	P3								B7
B8	IP								B9
C0									C7
C8									CF
D0	PSW								D7
D8									DF
E0	ACC								E7
E8									EF
F0	B								F7
F8									FF

Blue background are I/O port SFRs
Yellow background are control SFRs
Green background are other SFRs

Source: 8052.com

Grey boxes unusable addresses, registers in first column are bit addressable

TCON Register

TCON: TIMER/COUNTER CONTROL REGISTER. BIT ADDRESSABLE.

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 overflow flag. Set by hardware when the Timer/Counter 1 overflows. Cleared by hardware as processor vectors to the interrupt service routine.
TR1	TCON.6	Timer 1 run control bit. Set/cleared by software to turn Timer/Counter 1 ON/OFF.
TF0	TCON.5	Timer 0 overflow flag. Set by hardware when the Timer/Counter 0 overflows. Cleared by hardware as processor vectors to the service routine.
TR0	TCON.4	Timer 0 run control bit. Set/cleared by software to turn Timer/Counter 0 ON/OFF.
IE1	TCON.3	External Interrupt 1 edge flag. Set by hardware when External Interrupt edge is detected. Cleared by hardware when interrupt is processed.*
IT1	TCON.2	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.
IE0	TCON.1	External Interrupt 0 edge flag. Set by hardware when External Interrupt edge detected. Cleared by hardware when interrupt is processed.*
IT0	TCON.0	Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low level triggered External Interrupt.

***These flags cleared automatically by hardware only if edge triggered (otherwise have to be cleared by software)**

TMOD

TMOD: TIMER/COUNTER MODE CONTROL REGISTER. NOT BIT ADDRESSABLE.

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
Timer 1				Timer 0			

GATE	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/$\overline{T}$	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	Mode selector bit. (NOTE 1)
M0	Mode selector bit. (NOTE 1)

NOTE 1:

M1	M0		Operating Mode
0	0	0	13-bit Timer (8048 compatible)
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standart Timer 0 control bits. TH0 is an8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

Interrupts

INTERRUPTS:

To use any of the interrupts in the 80C51 Family, the following three steps must be taken.

1. **Set the EA (enable all) bit in the IE register to 1.**
2. **Set the corresponding individual interrupt enable bit in the IE register to 1.**
3. **Begin the interrupt service routine at the corresponding Vector Address of that interrupt. See Table below.**

INTERRUPT SOURCE	VECTOR ADDRESS
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI & TI	0023H

In addition, for external interrupts, pins INT0 and INT1 (P3.2 and P3.3) must be set to 1, and depending on whether the interrupt is to be level or transition activated, bits IT0 or IT1 in the TCON register may need to be set to 1.

ITx = 0 level activated

ITx = 1 transition activated

Interrupt Enable Register

IE: INTERRUPT ENABLE REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	–	–	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	Not implemented, reserved for future use.*
—	IE.5	Not implemented, reserved for future use.*
ES	IE.4	Enable or disable the serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External Interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

* User software should not write 1s to reserved bits. These bits may be used in future 80C51 products to invoke new features.

Interrupt Priority

ASSIGNING HIGHER PRIORITY TO ONE OR MORE INTERRUPTS:

In order to assign higher priority to an interrupt the corresponding bit in the IP register must be set to 1.

Remember that while an interrupt service is in progress, it cannot be interrupted by a lower or same level interrupt.

PRIORITY WITHIN LEVEL:

Priority within level is only to resolve simultaneous requests of the same priority level.

From high to low, interrupt sources are listed below:

IE0
TF0
IE1
TF1
RI or TI

IP: INTERRUPT PRIORITY REGISTER. BIT ADDRESSABLE.

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

–	–	–	PS	PT1	PX1	PT0	PX0

–	IP.7	Not implemented, reserved for future use.*
–	IP.6	Not implemented, reserved for future use.*
–	IP.5	Not implemented, reserved for future use.*
PS	IP.4	Defines the Serial Port interrupt priority level.
PT1	IP.3	Defines the Timer 1 interrupt priority level.
PX1	IP.2	Defines External Interrupt 1 priority level.
PT0	IP.1	Defines the Timer 0 interrupt priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level.

* User software should not write 1s to reserved bits. These bits may be used in future 80C51 products to invoke new features.

EE309 (Autumn 2013) – MidSem Solutions

Question 1:

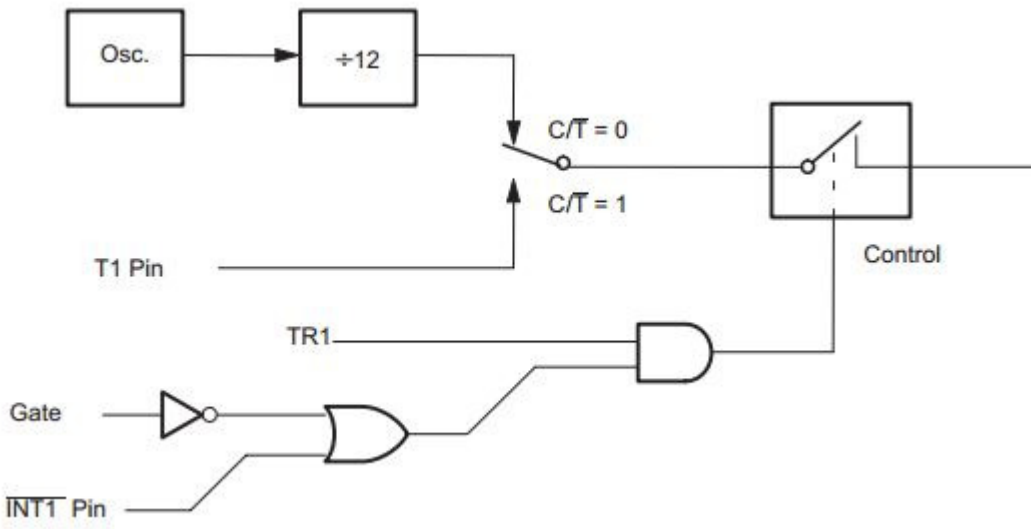
```
POP DPH          ; COPY HIGHER BYTE OF ADDRESS
POP DPL          ; COPY LOWER BYTE OF ADDRESS
CLR A            ;
JMP @A+DPTR      ; JUMP TO THE NEXT ADDRESS, FROM WHICH A CALL FUCTION
IS USED
```

Marking Scheme:

- 0.5 Mark if you have given idea that : the address is available in stack
- 1 Mark if you have taken the address using POP instructions
- 1.5 Mark if you have done the above and do some jump improperly
- 2 Mark if you have done all these steps without any logical errors

- **Notes :**
  - JUMP (with any fixed label) cannot replace RET function of any subroutine
  - No marks reduced for changing the order of first three instructions or syntax errors

Question 2:



- **Marking Scheme :**
  - 0.5 Mark if you have drawn the above diagram(or any other representations) with some sort of logical errors(or wrong marking of labels)
  - 1 Mark if you have drawn the above diagram (or any other representations) without any logical errors

Question 3:

```
SETB P3.4              ; make P3 as input
MOV TH0, 0xFC          ; #252
MOV TL0, 0x18          ; #24
TMOD 0xY5              ; lower nibble has to be 5 : GATE0=0, C/T_bar = 1, and Mode 1
SETB EA                ;  IE.7
SETB ET0               ;  IE.2
SETB  TR0              ; this should be the last instr
```

Order is not important

Marking Scheme :

- 7 instructions + 1 order = 8
  - Each couple of errors cost 0.5marks

4.
a) No. of T states = 12 (3(OF)+1(Decoding)+3(MR)+3(MW)+2(SP decrement))
(1mark for the correct answer)

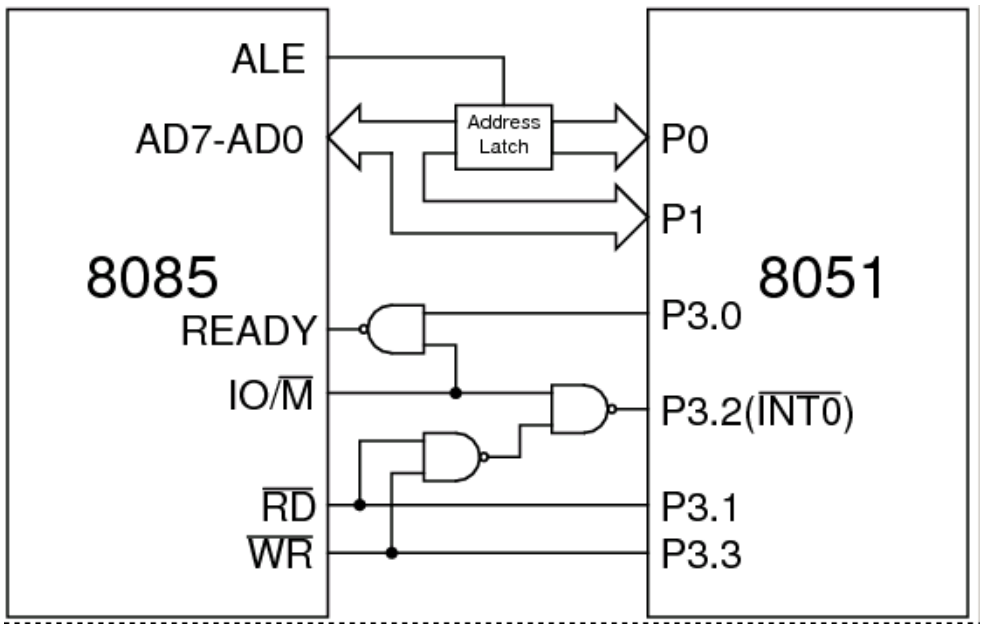
b) PUSH rp: The contents of the register pair are copied into the stack.
((SP)-1) ← (rh)
((SP)-2) ← (rl)
(SP) ← (SP)-2
This means, the stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that. Each decrement operation requires 1 T state and hence, 2 extra states are required in during execution of PUSH rp.
(full mark if you mention the reason as "SP decremented twice")

c) In POP rp, the contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1. But here, the increment happens in parallel to the copying operation and hence no extra T states are required. While in PUSH rp, decrement has to be done prior to copying and hence extra T states are required.
(full mark if you mention "SP increments in parallel")

5.

	A8-15	AD0-7	Marks allotment
Opcode Fetch	10	FF, CALL	1
Memory Read	11	00, 00	1.5
Memory Read	11	01, 40	
Memory Write	12	33, 11	1.5
Memory Write	12	32, 02	

6 a)



Marking Scheme:

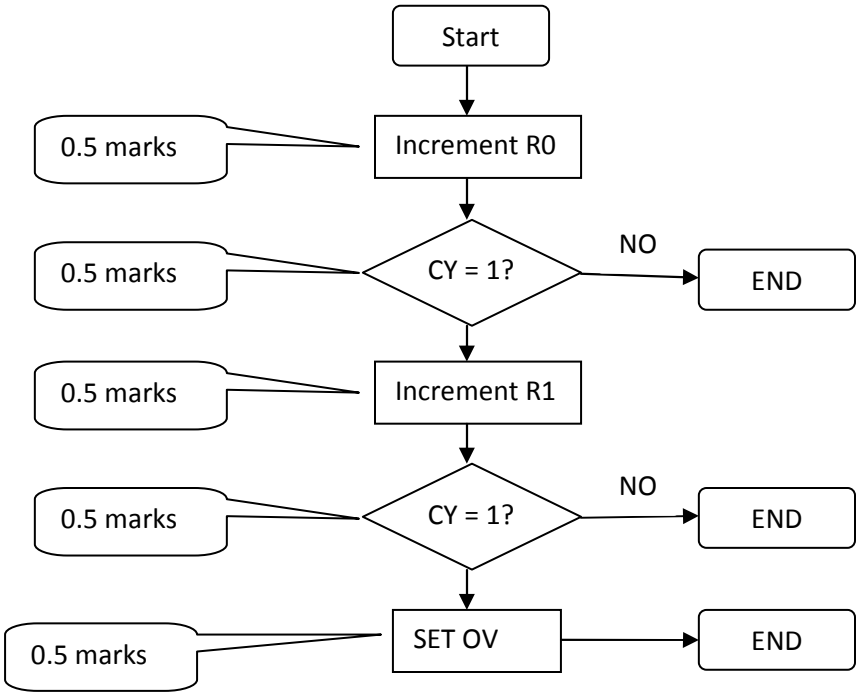
- Proper understanding of the problem and identification of handshake signals - 0.5 mark
- Identifying proper port(s) in 8051 for data transfer - 0.5 mark
- Generating proper interrupt - 0.5 mark
- Proper usage of READY, RD, WR and IO signals - 0.5 mark

6 b) Marking Scheme for a sample code:

```
; Initialization of ports in 8051 - 1 mark
; 0.5 mark if any one instruction is correct and 1 mark if all instructions are
correct
MOV P0, #0FFH
MOV P1, #0FFH
MOV P2, #0FFH

; Initialization of required bit addressable registers - 1 mark
; 0.5 mark if any one instruction is correct and 1 mark if all instructions are
correct
SETB EX0/1E.0
SETB IT0/TCON.0
SETB EA/IE.7

; ISR for read and write - 2 marks
MOV A,P0 ; getting proper RAM address - 0.5 mark
ANL A, #7FH
MOV R0, A

JNB P3.1, READ ; selection of read or write operation - 0.5 mark
JNB P3.3, WRITE
JMP DONE

READ: ; instructions for read operation - 0.5 mark
MOV P1, @R0
CLR P3.0
JMP DONE

WRITE: ; instructions for write operation - 0.5 mark
CLR P3.0
MOV @R0, P1

DONE:
MOV P0, #0FFH
MOV P1, #0FFH
MOV P2, #0FFH
RETI
```

Note: There is a special case if the address is zero itself (i.e. the address of R0). This case can be dealt with separately by first checking for this condition. If this condition is true, data can directly be moved between P2 and R0. Also, ideally the value in R0 should be stored in a temporary register such as B when this R0 is being used to store indirect transfer address. However, no one is being penalized if these cases are not considered.

Question 7: **Flow Chart**



0.5 marks for PUSH and POP of ACC.

Sample Code for Q7:

```
Subroutine_start:  PUSH A
                   MOV A, R0
                   ADD A, #01
                   DA A
                   MOV R0, A
                   JNC return
                   MOV A, R1
                   ADD A, #1
                   DA
                   MOV R1, A
                   JNC return
                   SETB OV
           return: POP A
                   RET
```

(Ideally OV should be cleared also in case there is no roll-over of the counter, but we're not giving/deducting marks for this as it was not explicitly mentioned.)

Question 8:

```
CLR C           ; Carry = 0
MOV A, DPL      ; DPL → A
SUBB A, #34H    ; A = A - #34H i.e. DPL - #34H
MOV DPL, A      ; A → DPL, result stored in DPL
MOV A, DPH      ; DPH → A
SUBB A, #12H    ; A = A - #12H i.e. DPH - #12H
MOV DPH, A      ; A → DPH, result stored in DPH
```

Question 9:

(a)

```
SETB P1.0       ; P1.0 latch ← 1 ;  P1.0 pin = 0 (due to external wire to gnd) ;

CPL P1.0        ; P1.0 latch = 0 (complement of last P1.0 latch value),

                ;since it is a read-port-write instruction

MOV C, P1.0     ; P1.0 pin → C i.e. C = 0 (read from pin)
```

State of Carry flag = 0 ; (0.5 Marks)
Reason: C gets P1.0 pin (=0), status of P1.0 latch doesn't matter ; (0.5 Marks)

(b) State of Carry flag = 0 ; (0.5 Marks)

Reason: P1.0 latch=0 after second instruction as in (a). Therefore, even if ground wire is removed, the pin status remains 0. Therefore, C ← P1.0 pin (=0) ; (0.5 Marks)