EE309 – Microprocessors

Solutions – Final Exam, Autumn 2013, IIT Bombay

November 19, 2013 (14:00 – 17:00 hrs) Maximum Marks: 40

In all questions requiring explanation, please be brief and to the point. Wrong statements can neutralize the credit obtained from correct statements.

1. Consider the following 8051 assembly level code (it is only a part of a complete program). *[7]*

```
 ORG 100H
100H  MOV P1, #0FFH
103H  MOV DPTR, #LABEL2
 LABEL1: MOV A, #6H
         ANL A, P1
         JMP @A+DPTR
 LABEL2: SJMP LABEL1
         AJMP 21FH
         LJMP 0922H
         LJMP 0F55H
```

   (a) Write the starting address of each of the assembly level instruction for the given code.

   (b) Write the hex code for these instructions – you can write *xx* for opcodes (and for the first byte of the AJMP instruction). Assume that the data is arranged in the Big-Endian format in the program memory.

   (c) What is the code trying to accomplish?

   (d) This code has a logical problem (error). Mention the problem and make changes to the code to resolve the problem (no need to write the entire code again, just mention the changes).

2. You have to generate 1 Hz clock at P1.0 pin of a 8051 micro-controller. The average frequency should be exactly 1Hz, but small period-to-period deviations are allowed. You can use only one timer, and the reference oscillator F_{CLK} to the chip is 12 MHz. *[4]*

   (a) Describe an easy way to accomplish this task.

   (b) Write the set of instructions for the same, also include initialization instructions. Write your assumptions, if any.

3. For an 8085 microprocessor, the instruction `SHLD <address16>` copies the contents of L register to the memory location specified by the operand and the contents of H register are stored into the next memory location. *[6]*

   (a) Give the sequence of byte values available at the A15-A8 pins and the AD7-AD0 pins while this instruction is fetched and executed. Assume the starting address where this instruction is located is 28FEH, address16=3456H, L contains 55H, and H contains 66H, opcode for SHLD is 22H.

   (b) Given that the instruction SHLD requires 2 T-states more than that predicted by the "T-States = 3n+1" rule. What is the value of n for SHLD and why is the said rule violated?

   (c) Briefly describe how the READY pin of the 8085 processor is used by slow memories to ensure that there are no errors during read/write operations.

4. (a) What is the lower limit on the baud-rate directly supported by the UART of the 8051 micro-controller for a given F_{CLK}.

(b) Can there be a workaround that allows us to overcome this limitation while still using the dedicated UART hardware of the microcontroller for serial data transfer (explain briefly)?

(c) What is the main advantage provided by the multi-processor mode of communication in the 8051 serial interface (when multiple processors have to be interfaced)? *[1+1+1]*

8051 Serial Port Control Bits:

Bit	Bit Name	Description
SCON.7	SM0	$0 \Rightarrow$ 8-bit data; $1 \Rightarrow$ 9-bit data
SCON.6	SM1	$0 \Rightarrow \left(\text{Baud-rate}=\frac{F_{clk}}{12} \text{ for SM0=0; } \frac{2^{SMOD}.F_{clk}}{64} \text{ for SM0=1}\right);$ $1 \Rightarrow \text{Baud-rate}=\frac{2^{SMOD}.F_{clk}}{12\times32\times(\text{No. of Timer1 machine cycles})}$
SCON.5	SM2	For multi-processor communication (If SM2=1, hardware asserts RI only if RB8=1, when serial data is received)
SCON.4	REN	Receive enable
SCON.3	TB8	9th transmit data bit in 9-bit UART mode
SCON.2	RB8	9th received data bit in 9-bit UART mode
SCON.1	TI	Transmit interrupt flag
SCON.0	RI	Receive interrupt flag
PCON.7	SMOD	Double baud-rate for {SM0,SM1} = {0,1}, {1,0} or {1,1}

5. (a) How many hexadecimal digits of precision is provided by a 8-digit decimal number (i.e. how many hexadecimal digits will be required to represent an 8-digit decimal number). The answer need not be an integer.

   (b) How will you represent the decimal number –5 (negative 5) as an 8-bit binary number in two's complement format? How is the same number represented as a 16-bit binary number in two's complement format? *[1+1]*

6. The Accumulator A of an 8051 microcontroller contains binary digits $(10000000)_2$. *[2]*

   (a) What is the value of contents of A (in decimal format) if the contents are treated as

      i. a binary number in two's complement format?

      ii. a BCD (binary coded decimal) number?

   (b) What will be the contents of A (binary digits) after "`DA A`" is executed, given that AC was 1, and C was 0 before just before execution?

7. Consider a 16-bit adder with two 16-bit binary inputs, carry-in input, one 16-bit binary output and one carry-out output. The 16-bit adder basically consists two identical 8-bit adders (with individual carry-in and carry-out bits). *[6]*

   (a) Neatly draw a sketch to show how the two 8-bit adders can be pipelined to double the throughput. Include all muxes, demuxes, flip-flops etc., wherever required and also label the clock periods etc. that go to different blocks.

   (b) The 16-bit adder requires $4\,ns$ for one computation without pipelining for a 5 V supply, and the supply voltage V_{DD} can be reduced with increasing computation time T_C using the formula $(V_{DD}-1)^2 \times T_C = C_1$, where V_{DD} is in Volts. Also, assume that the energy required per computation is $C_2V_{DD}^2$.

      i. By what factor can the power dissipation be reduced by pipelining (as in (a)) if the throughput of $4\,ns$ per 16-addition is maintained.

ii. By what factor can the computations be speeded-up by pipelining (as in (a)), if V_{DD} =5 V.

iii. For what value (expression) of V_{DD} is the energy-delay product minimum.

8. Consider a set of instructions for a pipelined MIPS processor having five pipeline stages: IF (instruction fetch), ID (instruction decode), EX (Execution), MEM (memory Reference), WB (write back), as discussed in the class. Instructions 1, 2 load registers R1 and R2, respectively. Instructions 3 and 4 calculate R3=R1+R2 and R4=R1+R5, respectively, where R5 already contains valid data. Instruction 5 is an unconditional jump instruction to a new address that is available during ID cycle itself). Instruction 6 at the new location stores the value of R4 into the memory. Assume that two registers can be read from and one register can be written to concurrently in the register bank. Also, if there is a concurrent read and write to the same register, the value read from the register is the updated value. *[6]*

Cycle No.→	1	2	3	4	5	6	7	8	9	10	..	..
`Instr. 1`	IF	ID	EX	M	WB							
`Instr. 2`												
`Instr. 3`												
`Instr. 4`												
..												
..												

(a) Show the timing for the instructions in the pipeline, if data forwarding is NOT used and data and instruction memories are shared (indicate stall cycles and corresponding hazards).

(b) Show the timing for the instructions in the pipeline, if data forwarding is NOT used but the data and instruction memories ARE SEPARATE (indicate stall cycles and corresponding hazards).

(c) Show the timing for the instructions in the pipeline, if data forwarding IS used, the data and instruction memories ARE SEPARATE, and the compiler can rearrange the instructions to minimize the hazards (indicate stall cycles and corresponding hazards). You can also assume that some instruction(s) can be moved after the unconditional branch instruction (as was suggested by one of the students in the class lecture).

9. For the different architectures listed in second row of columns 2, 3 and 4, fill out the boxes in the table, if the task to be accomplished is X3=(X1+X2)+(X1+X3), where X1, X2 and X3 represent the values stored at three different memory locations. Assume that the instruction & data memories are separate and the clock frequency is same for each processor. Also assume that each memory access requires around 10 stall cycles on an average (because there is no cache). *[4]*

Architecture	Stack based	Accumulator based	Register-Register (Load-Store)

Available Instruction Set	`PUSH Msrc`
	`POP Mdest`
	`ADD`
		`LOAD Msrc`
		`STORE Mdest`
		`ADD Msrc`
			`LOAD Rdest, Msrc`
			`STORE Rsrc, Mdest`
			`ADD Rdest, Rsrc1, Rsrc2`

Write a set of Instructions to obtain ~~M4=M1+M2+M3~~			
Compare the overall throughput of each (pipeline if possible)			

In this problem, the instructions PUSH, POP, LOAD, STORE etc. have their usual meaning.

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
DJNZ	Rn,rel	Decrement first operand and jump if not zero	24
	direct,rel		24
NOP		No Operation	12

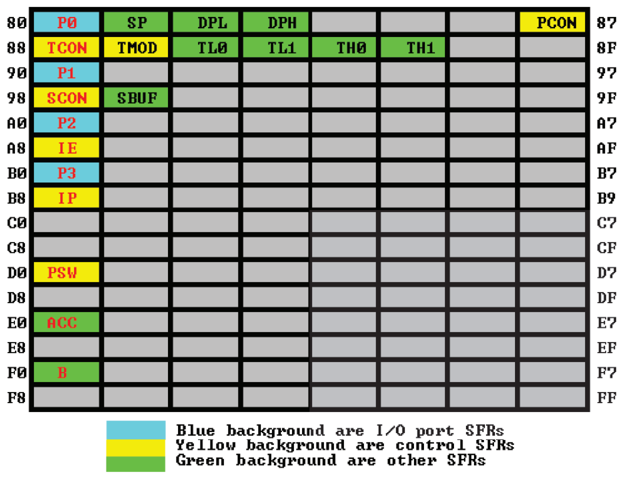
Instruction	Possible Operands	Description	Oscillator Periods
Data Transfer			
MOV	A,Rn A,direct A,@Ri A,#data Rn,A Rn,#data direct,A @Ri,A @Ri,#data	Move src to dest	12
MOV	Rn,direct direct,Rn direct,direct direct,@Ri direct,#data @Ri,direct DPTR,#data16	Move src to dest	24
MOVC	A,@A+DPTR A,@A+PC	Move Code byte to Accumulator	24
MOVX	A,@Ri A,@DPTR @Ri,A @DPTR,A	Move External RAM byte to Accumulator Move Accumulator contents to External RAM	24
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.

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.

Question 1

(a) Address	**(b)**Hex code	
		ORG 100H
0100H	75 90 FF	MOV P1, #0FFH
0103H	90 01 0B	MOV DPTR, #LABEL2
0106H	74 06	LABEL1: MOV A, #6H
0108H	55 90	ANL A, P1
010AH	73	JMP @A+DPTR
010BH	80 FB	LABEL2: SJMP LABEL1
010DH	41 1F	AJMP 21FH
010FH	02 09 22	LJMP 0922H
0112H	02 0F 55	LJMP 0F55H

(c) Code initializes port 1 as input port. Depending on value of P1.1 and P1.2, it tries to jump at one of the four locations i.e. 010BH, 010DH, 010FH, 0111H.

(d) Logical error:
Program tries to jump at 0111H, which is valid address but not starting address of any instruction. This results in undesired program outcome.

Changes:
```
ORG 100H
MOV P1, #0FFH
MOV DPTR, #LABEL2
LABEL1: MOV A, #6H
        ANL A, P1
        JMP @A+DPTR
LABEL2: SJMP LABEL1
        AJMP 21FH
        SJMP LABLE3
        LJMP 0F55H
LABLE3: LJMP 0922H
```

Marking scheme: (a) 2 Marks
(b) 2 Marks
(c) 1 Mark
(d) 2 Marks

Question 2

(a) Use timer in 8 bit auto reload mode (mode 2) and use 250 as the reload value. Timer generates interrupt after every 250us. Two registers are used to count the number of interrupts occurred. When the count reaches 2000 (i.e. 0.5s) P1.0 is complemented. **(2 marks)**

(b) Sample code for generating 1Hz clock using mode 2 of timer 0: **(2 marks)**

Initialization: **(1 mark)**

```
MOV  R1,      #32H      ; count value of 50
MOV  R2,      #28H      ; count value of 40. R1*R2 = 2000

MOV  TH0,     #06H      ; TH0 & TL0 initialized for a count of 250
MOV  TL0,     #06H
MOV  TMOD,    #02H      ; timer 0 auto reload mode

SETB P1.0               ; initial level of clock assumed to be high

SETB TR0                ; turn timer 0 ON
SETB ET0                ; enable timer 0 overflow interrupt
SETB EA                 ; global interrupt enable
```

Interrupt Service Routine: **(1 mark)**

```
    DJNZ R1,      ISR_RET   ; decrement R1, RETI if R1 != 0
    MOV  R1,      #32H      ; reload count value of 50

    DJNZ R2,      ISR_RET   ; decrement R2, RETI if R2 != 0
    MOV  R2,      #28H      ; reload count value of 40

    CPL  P1.0               ; complements P1.0 after 0.5s

ISR_RET:

    RETI                    ; return from ISR
```

Question 3

(a) Sequence of hex values available at the A15-A8 and AD7-AD0 bus: **(4 marks)**
(1 mark each for any 3 correct memory access cycle, full marks if all 5 cycles are correct)

A15-A8	**AD7-AD0**
28H	FEH
	22H
28H	FFH
	56H
29H	00H
	34H
34H	56H
	55H
34H	57H
	66H

(b) 'n' is the number of memory accesses needed in the instruction. For SHLD n = 5.
(0.5 marks)

If "T-States = 3n+1" is followed, SHLD would require 16 T-States for execution. If it is given that SHLD requires two additional T-states, these states must be required for incrementing the 16-bit memory address to get the address to which H has to be copied **(0.5 marks)**

(c) Logic HIGH on READY pin tells the microprocessor that the device with which it is communicating is ready for read/write operation. A slow memory will de-assert READY signal as soon as it is addressed and asserts it once data is written to the memory from A/D bus or data is read to A/D bus from the memory. **(0.5 marks)**

The microprocessor samples READY signal on the rising edge of the clock after falling edge of ALE signal and repeats sampling on rise edges of the clock until READY is asserted by the memory. Once the memory acknowledges successful data transfer by asserting READY signal, microprocessor starts using A/D bus for other operations, thereby ensuring error free read/write operations. **(0.5 marks)**

Question 4

(a) Lower limit on baud rate:

$$Baud\ rate = \frac{2^{SMOD}.F_{CLK}}{12 \times 32 \times 65536}$$

(b) This limitation can be overcome by avoiding Timer 1 overflow. For this, Timer 0 can be used to reset Timer 1 before it overflows.

(c) Main advantage:

In multiprocessor mode, the main advantage is that the Master can communicate with a certain slave without interrupting other slaves (when the SM2 bit in SCON register is used).

Question 5(a)

$$Number\ of\ hexadecimal\ digits = \log_{16}(10^8) = 6.644$$

1 Mark

Question 5(b)

8 bits	---------------------11111011	0.5Mark
16 bits	--------1111111111111011	0.5Mark

Question 6(a)

i)	-128	0.5 Mark
ii)	80	0.5 Mark

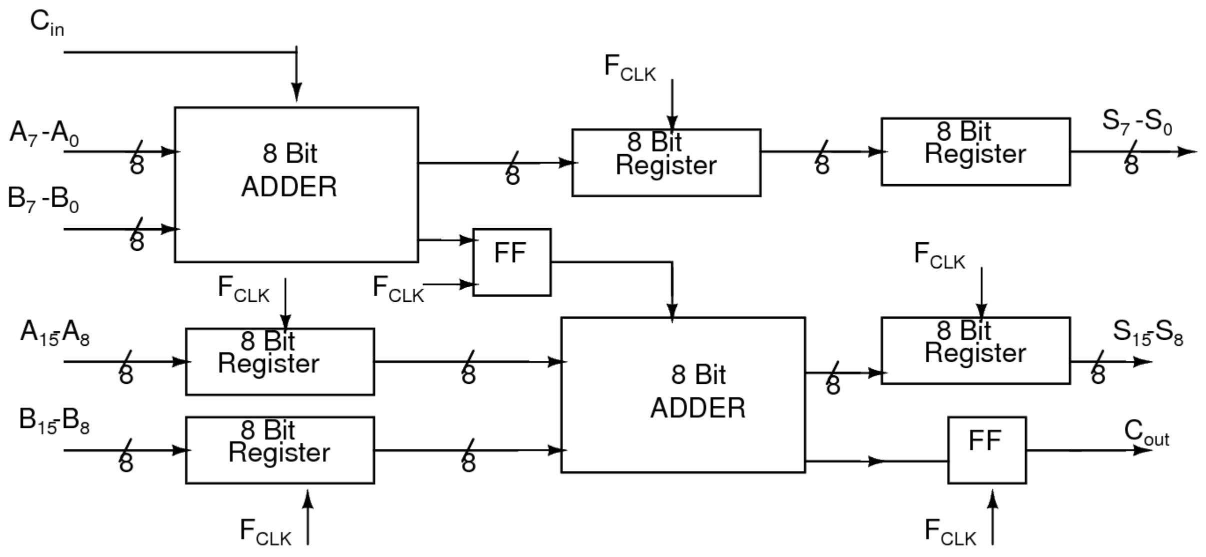
Question 6(b)

Ans: $(10000110)_{BCD} = (86)_{10}$ 1 Mark

[AC=1, so $(0110)_2$ has to be added to the lower nibble]

Question 7(a)

Mark distribution:
1 Mark: Implementation of 16 bit adder
1 Mark: Pipelining (by using registers/flip flops)
1 Mark: Proper labelling of all wires/buses



Question 7(b)

1 Mark each

i)

$$(V_{DD_{new}} - 1)^2 \times T_{C_{new}} = (V_{DD_{old}} - 1)^2 \times T_{C_{old}}$$

$$(V_{DD_{new}} - 1)^2 \times (8\,ns) = (5-1)^2 \times (4\,ns)$$

$$V_{DD_{new}} = 3.828\,V$$

$$\frac{P_{new}}{P_{old}} = \left(\frac{V_{DD_{new}}}{V_{DD_{old}}}\right)^2 = 0.586\,V$$

ii) 2

iii) $V_{DD} > 1$ V; $Energy * Delay = C_1 C_2 \frac{V_{DD}^{\;2}}{V_{DD} - 1}$

For minimum Energy-Delay Product: $V_{DD} \rightarrow Infinity$

Question 8: There can be different solutions based on assumptions of how the pipeline is implemented. As long as your implementation is conceptually not wrong, you get full credit.

Q8.(a) - Independent hazards

	1	2	3	4	5	6	7	8	9	10	11	12	13
LD R1	IF	ID	EX	M	WB								
LD R2		IF	ID	EX	M	WB							
R3 = R1+R2			IF	ID	**DH**	EX	-	WB					
R4 = R1+R5				**SH**	**SH**	IF	ID	EX	-	WB			
JMP <#add>							IF	ID	-	-	-		
								IF/**CH**					
STR R4									IF	ID	EX	M	-

Marking Scheme: Finding each hazard carries 0.5 Mark → total 2 Marks

Q8.(b) - Independent hazards

	1	2	3	4	5	6	7	8	9	10	11	12	13
LD R1	IF	ID	EX	M	WB								
LD R2		IF	ID	EX	M	WB							
R3 = R1+R2			IF	ID	**DH**	EX	-	WB					
R4 = R1+R5				**IF**	**ID**	**SH**	EX	-	WB				
JMP <#add>					IF	ID	-	-	-				
						IF/**CH**							
STR R4							IF	ID	EX	M	-		

Marking Scheme: Finding each hazard carries 0.5 Mark → total 1.5 Marks

Q8.(c) - Independent hazards

	1	2	3	4	5	6	7	8	9	10	11	12	13
LD R1	IF	ID	EX	M	WB								
LD R2		IF	ID	EX	M	WB							
R4 = R1+R5			IF	ID	**DH**	EX	-	WB					
JMP <#add>				IF	ID								
R3 = R1+R2					IF	ID	EX	-	WB				
STR R4						IF	ID	EX	M	-			

Marking Scheme:
Each re-ordering carries 1 → 2 Marks
Finding hazard carries 0.5 Mark → total 0.5 Marks

DH : Data Hazards
SH : Structural Hazards
CH : Control Hazards

<u>Question 8:</u> Another possible implementation

Q8.(a) - Stall

	1	2	3	4	5	6	7	8	9	10	11	12	13
LD R1	IF	ID	EX	M	WB								
LD R2		IF	ID	EX	M	WB							
R3 = R1+R2			IF	ID	**S1**	EX	-	WB					
R4 = R1+R5					**S1**	IF	ID	EX	-	WB			
JMP <#add>							IF	ID	-	-	-		
								IF/**S2**					
STR R4									IF	ID	EX	M	-

Marking Scheme: Finding each hazard with details carries 1 mark

Q8.(b) - Stall

	1	2	3	4	5	6	7	8	9	10	11	12	13
LD R1	IF	ID	EX	M	WB								
LD R2		IF	ID	EX	M	WB							
R3 = R1+R2			IF	ID	**S3**	EX	-	WB					
R4 = R1+R5				**IF**	**ID**	**S4**	EX	-	WB				
JMP <#add>					IF	ID	-	-	-				
						IF/**S5**							
STR R4							IF	ID	EX	M	-		

Marking Scheme: Finding each hazard carries 0.5 Mark → total 1.5 Marks

Q8.(c) - Stall

	1	2	3	4	5	6	7	8	9	10	11	12	13
LD R1	IF	ID	EX	M	WB								
LD R2		IF	ID	EX	M	WB							
R3 = R1+R2			IF	ID	**S4**	EX	-	WB					
JMP <#add>				IF	ID								
R4 = R1+R5					IF	ID	EX	-	WB				
STR R4						IF	ID	EX	M	-			

Marking Scheme: Proper re-ordering carries 1 Mark
Indicating proper data-forward carries 1 Mark
Finding hazard carries 0.5 Mark

S1, S3 : Data Hazards
S2, S5 : Control Hazards
S4 : Structural Hazards

Question 9:

Architecture	Stack based **(1 Mark)**	Accumulator based **(1 Mark)**	Register-Register **(1 Mark)**
Instructions for evaluating X3=(X1+X2)+(X1+X3)	**PUSH X1 PUSH X2 ADD PUSH X1 PUSH X3 ADD ADD POP X3**	**LOAD X1 ADD X2 STORE X4 LOAD X1 ADD X3 ADD X4 STORE X3**	**LOAD R1, X1 LOAD R2, X2 LOAD R3, X3 ADD R4, X1, X2 ADD R5, X1, X3 ADD R3, R4, R5 STORE X3, R3**

Comparison between architectures: **(1 Mark for correct comparison)**

Stack based: Uses 5 memory references, pipelining is not possible.

Accumulator based: 7 memory references, pipelining can only help a little bit (even if implemented with great difficulty).

Register-Register architecture: 4 memory references, pipelining can also be easily implemented

Since each memory access results in significant number of stall cycles, the architectures that require less number of memory accesses are faster.

In this case, Register-Register architecture will have highest throughput, stack based architecture will have moderate throughput and Accumulator based architecture will have least throughput.