B.Sc. in EEE, 7th Semester October 06, 2023 **(Afternoon)**

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
ORGANISATION OF ISLAMIC COOPERATION (OIC)
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Mid-Semester Examination
Course No.: EEE 4705
Course Title: Microcontroller Based System Design

Winter Semester, A. Y. 2022-2023
Time: 90 Minutes
Full Marks: 75

There are 3 **(three)** questions. Answer all 3 **(three)** questions. The symbols have their usual meanings. Programmable calculators are not allowed. Marks of each question and corresponding COs and POs are written in the brackets.

---

1. a) Describe the criteria for choosing a microcontroller. **07 (CO1, PO1)**

   b) Design an 8 bit hexadecimal to decimal converter using the process of repeated division. Assume the 8 bit hexadecimal number is present in some RAM locations. You need to convert these numbers to decimal and save the result in some RAM locations of your choice. **18 (CO2, PO2)**

2. a) Interpret bank 1 and stack conflict. **07 (CO1, PO1)**

   b) Write a complete program using assembly language to find the weight of the modulo-2 sum of the number obtained from port 1 and a number in R2. **18 (CO2, PO2)**
   [Hint: Modulo-2 addition is a XOR operation on two binary numbers. 'Weight' means the number of 1s in it. Here it implies the number of bit positions in which the two numbers are different. Doing XOR operation on two numbers, in the following example, we get 1110. This is the modulo-2 sum and the weight of the sum is 3.]

   e.g.
   ```
    1101
    0011
   ------
    1110
   ```

3. a) Assuming that ROM space starting at 350H contains "Jurgen", write a program to transfer the first three bytes into R0, R1 and R2 of Bank-0 and the next three bytes to R0, R1 and R2 of Bank-1. **10 (CO2, PO2)**

   b) Generate a delay of 5 ms for an AT89C51 with a crystal frequency of your choice. Show necessary calculation. **15 (CO2, PO2)**

8051 Instruction Set Summary

Rn Register R7-R0 of the currently selected Register Bank.
Data 8-bit internal data location's address. This could be an internal Data RAM location (0-127) or a SFR [i.e. I/O port, control register, status register, etc. (128-255)].
@Ri 8-bit Internal Data RAM location (0-255) addressed indirectly through register R1 or R0.
#data 8-bit constant included in instruction.
#data16 16-bit constant included in instruction.
addr16 16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the 64k byte Program Memory address space.
addr11 11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2k byte page of Program Memory as the first byte of the following instruction.
rel Signed (two's component) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is –128 to +127 bytes relative to first byte of the following instruction.
bit Direct Addressed bit in Internal Data RAM or Special Function Register.

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Note that operations on SFR byte address 206 or bit addresses 209-215 (i.e. the PSW or bits in the PSW) will also affect flag settings.

Mnemonic	Description	Byte	Cycle
Arithmetic operations			
ADD A,Rn	Add register to accumulator	1	1
ADD A,direct	Add direct byte to accumulator	2	1
ADD A,@Ri	Add indirect RAM to accumulator	1	1
ADD A,#data	Add immediate data to accumulator	2	1
ADDC A,Rn	Add register to accumulator with carry flag	1	1
ADDC A,direct	Add direct byte to A with carry flag	2	1
ADDC A,@Ri	Add indirect RAM to A with carry flag	1	1
ADDC A,#data	Add immediate data to A with carry flag	2	1
SUBB A,Rn	Subtract register to accumulator with borrow	1	1
SUBB A,direct	Subtract direct byte to A with carry borrow	2	1
SUBB A,@Ri	Subtract indirect RAM to A with carry borrow	1	1
SUBB A,#data	Subtract immediate data to A with carry borrow	2	1
INC A	Increment accumulator	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	1
INC @Ri	Increment indirect RAM	1	1
DEC A	Decrement accumulator	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	1
DEC @Ri	Decrement indirect RAM	1	1
INC DPTR	Increment data pointer	1	2
MUL AB	Multiply A and B -> [B hi]:[A lo]	1	4
DIV AB	Divide A by B -> A=result, B=remainder	1	4
DA A	Decimal adjust accumulator	1	1
CLR A	Clear accumulator	1	1
CPL A	Complement accumulator	1	1
RL A	Rotate accumulator left	1	1
RLC A	Rotate accumulator left through carry	1	1
RR A	Rotate accumulator right	1	1
RRC A	Rotate accumulator right through carry	1	1
SWAP A	Swap nibbles within the accumulator	1	1
Logic operations			
ANL A,Rn	AND register to accumulator	1	1
ANL A,direct	AND direct byte to accumulator	2	1
ANL A,@Ri	AND indirect RAM to accumulator	1	1
ANL A,#data	AND immediate data to accumulator	2	1
ANL direct,A	AND accumulator to direct byte	2	1
ANL direct,#data	AND immediate data to direct byte	3	2
ORL A,Rn	OR register to accumulator	1	1
ORL A,direct	OR direct byte to accumulator	2	1
ORL A,@Ri	OR indirect RAM to accumulator	1	1
ORL A,#data	OR immediate data to accumulator	2	1
ORL direct,A	OR accumulator to direct byte	2	1
ORL direct,#data	OR immediate data to direct byte	3	2
XRL A,Rn	Exclusive OR register to accumulator	1	1
XRL A,direct	Exclusive OR direct byte to accumulator	2	1
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	1	1
XRL A,#data	Exclusive OR immediate data to accumulator	2	1
XRL direct,A	Exclusive OR accumulator to direct byte	2	1
XRL direct,#data	Exclusive OR immediate data to direct byte	3	2
Boolean variable manipulation			
CLR C	Clear carry flag	1	1
CLR bit	Clear direct bit	2	1
SETB C	Set carry flag	1	1
SETB bit	Set direct bit	2	1
CPL C	Complement carry flag	1	1
CPL bit	Complement direct bit	2	1
ANL C,bit	AND direct bit to carry flag	2	2
ANL C,/bit	AND complement of direct bit to carry	2	2
ORL C,bit	OR direct bit to carry flag	2	2
ORL C,/bit	OR complement of direct bit to carry	2	2
MOV C,bit	Move direct bit to carry flag	2	1
MOV bit,C	Move carry flag to direct bit	2	2
Program and machine control			
ACALL addr11	Absolute subroutine call	2	2
LCALL addr16	Long subroutine call	3	2
RET	Return from subroutine	1	2
RETI	Return from interrupt	1	2
AJMP addr11	Absolute jump	2	2
LJMP addr16	Long jump	3	2
SJMP rel	Short jump (relative address)	2	2
JMP @A+DPTR	Jump indirect relative to the DPTR	1	2
JZ rel	Jump if accumulator is zero	2	2
JNZ rel	Jump if accumulator is not zero	2	2
JC rel	Jump if carry flag is set	2	2
JNC rel	Jump if carry flag is not set	2	2
JB bit,rel	Jump if bit is set	3	2
JNB bit,rel	Jump if bit is not set	3	2
JBC bit,rel	Jump if direct bit is set and clear bit	3	2
CJNE A,direct,rel	Compare direct byte to A and jump if not equal	3	2
CJNE A,#data,rel	Compare immediate to A and jump if not equal	3	2
CJNE Rn,#data,rel	Compare immed. to reg. and jump if not equal	3	2
CJNE @Rn,#data,rel	Compare immed. to ind. and jump if not equal	3	2
DJNZ Rn,rel	Decrement register and jump in not zero	2	2
DJNZ direct,rel	Decrement direct byte and jump in not zero	3	2
NOP	No operation	1	1
Data transfer			
MOV A,Rn	Move register to accumulator	1	1
MOV A,direct*)	Move direct byte to accumulator	2	1
MOV A,@Ri	Move indirect RAM to accumulator	1	1
MOV A,#data	Move immediate data to accumulator	2	1
MOV Rn,A	Move accumulator to register	1	1
MOV Rn,direct	Move direct byte to register	2	2
MOV Rn,#data	Move immediate data to register	2	1
MOV direct,A	Move accumulator to direct byte	2	1
MOV direct,Rn	Move register to direct byte	2	2
MOV direct,direct	Move direct byte to direct byte	3	2
MOV direct,@Ri	Move indirect RAM to direct byte	2	2
MOV direct,#data	Move immediate data to direct byte	3	2
MOV @Ri,A	Move accumulator to indirect RAM	1	1
MOV @Ri,direct	Move direct byte to indirect RAM	2	2
MOV @Ri,#data	Move immediate data to indirect RAM	2	1
MOV DPTR,#data16	Load data pointer with a 16-bit constant	3	2
MOVC A,@A+DPTR	Move code byte relative to DPTR to accumulator	1	2
MOVC A,@A+PC	Move code byte relative to PC to accumulator	1	2
MOVX A,@Ri	Move external RAM (8-bit addr.) to A	1	2
MOVX A,@DPTR	Move external RAM (16-bit addr.) to A	1	2
MOVX @Ri,A	Move A to external RAM (8-bit addr.)	1	2
MOVX @DPTR,A	Move A to external RAM (16-bit addr.)	1	2
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A,Rn	Exchange register to accumulator	1	1
XCH A,direct	Exchange direct byte to accumulator	2	1
XCH A,@Ri	Exchange indirect RAM to accumulator	1	1
XCHD A,@Ri	Exchange low-order nibble indir. RAM with A	1	1

*) MOV A,ACC is not a valid instruction

jne A,#data,@ — cjne A,#data,@
(jump if A ! = data)

je A, #data,@ — add A,#low(data) *or* cjne A,#(data),ne
(jump if A == data) — jz @ — jmp @ — ne: ...

ja, jnbe A,#data,@ — add A,#low(data 1) *or* cjne A,#(data+1),ne
(jump if A > data) — jc @ — jnc @

jae, jnb A,#data,@ — add A,#low(data) *or* cjne A,#(data),ne
(jump if A >= data) — jc @ — jnc @

jb, jnae A,#data,@ — add A,#low,(data) *or* cjne A,#(data),ne
(jump if A < data) — jnc @ — jc @

jbe, jna A,#data,@ — add A,#low(data 1) *or* cjne A,#(data+1),ne
(jump if A <= data) — jnc @ — jc @

switch A <,==,> #data — cjne A,#data,ne
(no A modification) — ... — ; execute code if A==data
ne: jc is_below — ; jump if A<data
jnc is_above — ; jump if A>data or exec. code

This paper was created by Št pán Mat jka alias Mates for anybody who needs it. Mates, Prague – Czech Republic 1998,2002.