Microprocessor Applications in Manufacturing – MEL432
Major Exams

Date: 07 May 2007
Time: 13:00 pm to 15:00pm
Duration: Two hours
Maximum Marks: 60

Do any five questions. All Questions carry equal marks. **While writing programs,** kindly **write comments**, to make your program easier to understand.

Q1. a) In order to measure a maximum of 4V with a resolution of 1 mV, what should be the number of bits in the A/D converter? (2)

b) Define the terms accuracy and monotonicity with respect to an A/D converter. Why is monotonicity desirable? (3)

b) Explain the successive approximation method of A/D conversion. (4)

a) What is a sample and hold amplifier? (3)

Q2. a) What are microcontrollers? (3)

b) Describe some common features available on single chip microcontrollers. (5)

c) List some areas of their applications. (4)

Q3. Write short notes on **any two** of the following:

a) Special Function Registers of 8751. (6)

b) Serial Port of 8751 (6)

c) The Internal Data Memory of 8751 (6)

d) Stand-alone 8751 system. (6)

Q4. Write on **any two** of the following:

a) Multiplexing of 7-segment displays. (6)

b) Interfacing of keys. (6)

c) Asynchronous serial transmission of the letter A, (41H in ASCII) at 19200 baud. (Show a sketch). Show DTE to DTE three wire connection. (6)

d) Compare a computer monitoring system, a computer open loop system and a computer close loop system. (6)

Q5. Write short notes on **any three** of the following:

a) Tri-state concept, buffer registers and bus organized computers. (4)

b) Polling of Interrupts. (4)

c) Power–on Reset (using RESETIN* pin) (4)

d) Floating point number representation of binary numbers (give format). (4)

Q6. a) What is a stepper motor? How does it run? (4)

b) Write a program for running a stepper motor at 10 rpm using four bits of Port P2 of 8751. The motor takes 200 steps per revolution. The crystal frequency of 8751 is 12 MHz. The direction of rotation of the stepper motor is to depend on whether the Port Bit P1.0 (connected to a switch SPDT), is '0' or '1'. (No need to debounce the switch). Write a delay subroutine. (8)

<u>OR</u>

b) Write a program for running a stepper motor at 10 rpm using four bits of Port A of 8255 connected to a 8085. The motor takes 200 steps per revolution. The crystal frequency of 8085 is 6.144 MHz. The direction of rotation of the stepper motor is to depend on whether the bit 0 of Port C (connected to a switch SPDT), is '0' or '1'. Write the delay subroutine. (No need to debounce the switch) (8)

Q7. a) Explain the addressing modes of 8751 with examples. (4)

b) Explain how the indexed mode MOVC can be used for table look-up. (2)

c) Write **short segments** of programs or subroutines for **<u>any three</u>** of the following, for the 8051 microcontroller:

i) Change from Register Bank 0 to Register Bank 3 and set a Flag F0 located in PSW to '1'. (2)

ii) Divide a byte PQ at 30h to separate it into a byte 0Q at 31h and OP at 32h. (2)

iii) Make P1 into an input port. Input the port data at P1 and check P1.0. If it is '1' then output AA on LEDs connected to port P2. (2)

iv) Clear 16 memory locations from 31h onwards, using register R1 for indirect addressing. (2)

Q8. a) Write a program for 8751 to convert a binary number (less than 64H) stored at 30h into a BCD number at 31h. (Hint: divide by 10 Dec) (6)

b) Write a program to output BCD counting at pins of port P1 of 8751 with a delay of approx. 0.13 millisec. (XTAL Freq of 12 Mhz, gives an instruction timing of 1 μsec.) (6)

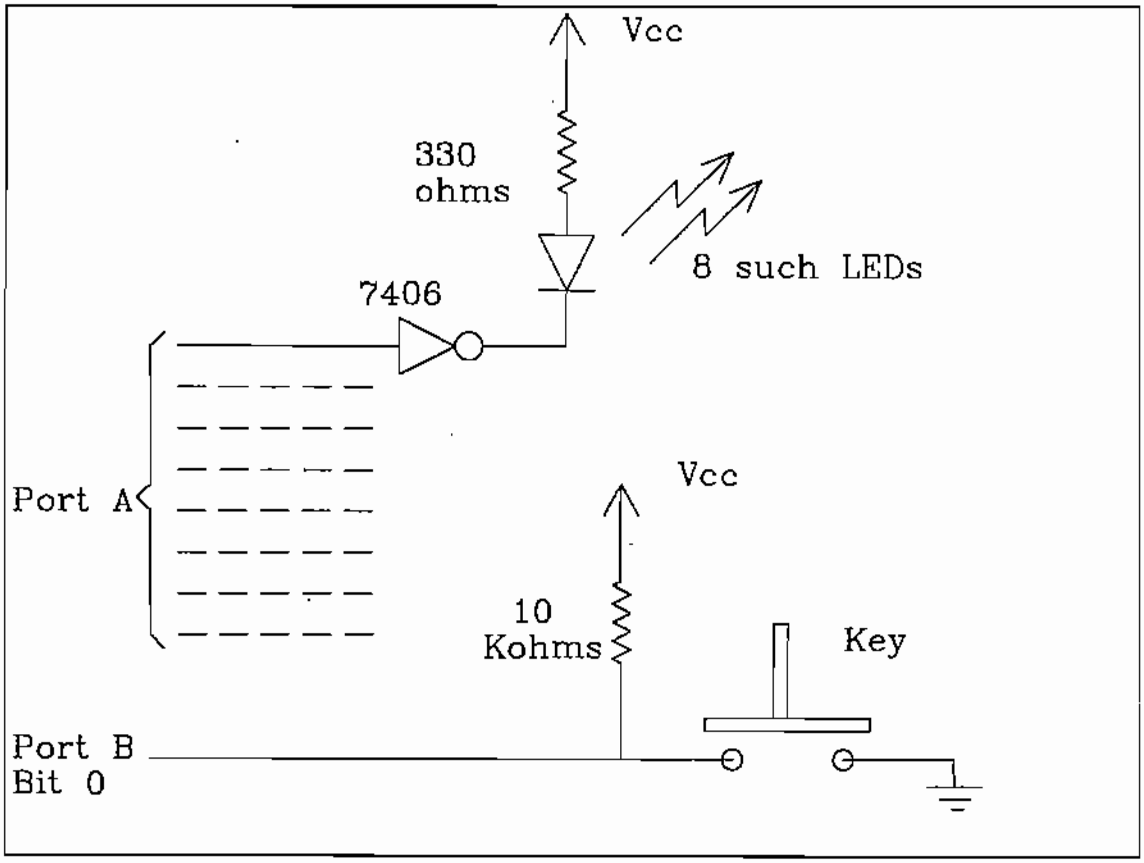
Q9. a) Describe a programmable peripheral interface, 8255. Explain the format for setting of the Control word, as well as bit set & reset function of Port C. (4)

b) What is bouncing of keys? (2)

c) Write a program to depict decimal counting on the LED's. The decimal count increments by one, every time a key is pressed. The LED's are connected to Port A. A key is connected to Port B, bit 0 of an 8255 as shown in Figure 1. Initialize the 8255 before using it. (6)

****

Figure 1.


Vcc
330
ohms
8 such LEDs
7406
Port A
Vcc
10
Kohms
Key
Port B
Bit 0

DATA TRANSFER GROUP

Move

MOV	A,A	7F	4
	A,B	78	
	A,C	79	
	A,D	7A	
	A,E	7B	
	A,H	7C	
	A,L	7D	
	A,M	7E	7
MOV	B,A	47	4
	B,B	40	
	B,C	41	
	B,D	42	
	B,E	43	
	B,H	44	
	B,L	45	
	B,M	46	7
MOV	C,A	4F	4
	C,B	48	
	C,C	49	
	C,D	4A	
	C,E	4B	
	C,H	4C	
	C,L	4D	
	C,M	4E	7
MOV	D,A	57	4
	D,B	50	
	D,C	51	
	D,D	52	
	D,E	53	
	D,H	54	
	D,L	55	
	D,M	56	7

Move (cont)

MOV	E,A	5F	4
	E,B	58	
	E,C	59	
	E,D	5A	
	E,E	5B	
	E,H	5C	
	E,L	5D	
	E,M	5E	7
MOV	H,A	67	4
	H,B	60	
	H,C	61	
	H,D	62	
	H,E	63	
	H,H	64	
	H,L	65	
	H,M	66	7
MOV	L,A	6F	4
	L,B	68	
	L,C	69	
	L,D	6A	
	L,E	6B	
	L,H	6C	
	L,L	6D	
	L,M	6E	7
MOV	M,A	77	7
	M,B	70	
	M,C	71	
	M,D	72	
	M,E	73	
	M,H	74	
	M,L	75	

XCHG	EB	4

Move Immediate

MVI	A, byte	3E	7
	B, byte	06	
	C, byte	0E	
	D, byte	16	
	E, byte	1E	
	H, byte	26	
	L, byte	2E	
	M, byte	36	10

Load Immediate

LXI	B, dble	01	10
	D, dble	11	
	H, dble	21	
	SP, dble	31	

Load/Store

LDAX B	0A	17
LDAX D	1A	
LHLD adr	2A	16
LDA adr	3A	13
STAX B	02	7
STAX D	12	
SHLD adr	22	16
STA adr	32	13

byte = constant, or logical/arithmetic expression that evaluates to an 8-bit data quantity. (Second byte of 2-byte instructions).

dble = constant, or logical/arithmetic expression that evaluates to a 16-bit data quantity. (Second and Third bytes of 3-byte instructions).

adr = 16-bit address (Second and Third bytes of 3-byte instructions).

* = all flags (C, Z, S, P, AC) affected.

** = all flags except CARRY affected; (exception: INX and DCX affect no flags).

† = only CARRY affected.

ARITHMETIC AND LOGICAL GROUP

Add*

ADD	A	87	4
	B	80	
	C	81	
	D	82	
	E	83	
	H	84	
	L	85	
	M	86	7
ADC	A	8F	4
	B	88	
	C	89	
	D	8A	
	E	8B	
	H	8C	
	L	8D	
	M	8E	7

Subtract*

SUB	A	97	4
	B	90	
	C	91	
	D	92	
	E	93	
	H	94	
	L	95	
	M	96	7
SBB	A	9F	4
	B	98	
	C	99	
	D	9A	
	E	9B	
	H	9C	
	L	9D	
	M	9E	7

Double Add †

DAD	B	09	10
	D	19	
	H	29	
	SP	39	

Increment**

INR	A	3C	4
	B	04	
	C	0C	
	D	14	
	E	1C	
	H	24	
	L	2C	
	M	34	10
INX	B	03	6
	D	13	
	H	23	
	SP	33	

Decrement**

DCR	A	3D	4
	B	05	
	C	0D	
	D	15	
	E	1D	
	H	25	
	L	2D	
	M	35	10
DCX	B	0B	6
	D	1B	
	H	2B	
	SP	3B	

Specials

DAA*	27	4
CMA	2F	
STC†	37	
CMC†	3F	

Rotate †

RLC	07	4
RRC	0F	
RAL	17	
RAR	1F	

Logical*

ANA	A	A7	4
	B	A0	
	C	A1	
	D	A2	
	E	A3	
	H	A4	
	L	A5	
	M	A6	7
XRA	A	AF	4
	B	A8	
	C	A9	
	D	AA	
	E	AB	
	H	AC	
	L	AD	
	M	AE	7
ORA	A	B7	4
	B	B0	
	C	B1	
	D	B2	
	E	B3	
	H	B4	
	L	B5	
	M	B6	7
CMP	A	BF	4
	B	B8	
	C	B9	
	D	BA	
	E	BB	
	H	BC	
	L	BD	
	M	BE	7

Arith & Logical Immediate

ADI byte	C6	7
ACI byte	CE	
SUI byte	D6	
SBI byte	DE	
ANI byte	E6	
XRI byte	EE	
ORI byte	F6	
CPI byte	FE	

RANCH CONTROL GROUP

Jump

JMP adr	C3	10
JNZ adr	C2	7/10
JZ adr	CA	
JNC adr	D2	
JC adr	DA	
JPO adr	E2	
JPE adr	EA	
JP adr	F2	
JM adr	FA	
PCHL	E9	

Call

CALL adr	CD	18
CNZ adr	C4	9/18
CZ adr	CC	
CNC adr	D4	
CC adr	DC	
CPO adr	E4	
CPE adr	EC	
CP adr	F4	
CM adr	FC	

Return

RET	C9	10
RNZ	C0	6/12
RZ	C8	
RNC	D0	
RC	D8	
RPO	E0	
RPE	E8	
RP	F0	
RM	F8	

Restart

RST	0	C7	12
	1	CF	
	2	D7	
	3	DF	
	4	E7	
	5	EF	
	6	F7	
	7	FF	

I/O AND MACHINE CONTROL

Stack Ops

PUSH	B	C5	12
	D	D5	
	H	E5	
	PSW	F5	
POP	B	C1	10
	D	D1	
	H	E1	
	PSW*	F1	

XTHL	E3	16
SPHL	F9	6

Input Output

OUT byte	D3	10
IN byte	DB	10

Control

DI	F3	4
EI	FB	4
NOP	00	4
HLT	76	4

New Instructions (8085 Only)

RIM	20	4
SIM	30	4

ASSEMBLER REFERENCE

Operators

( )
NUL
LOW, HIGH
*, /, MOD, SHL, SHR
+, −
NOT
AND
OR, XOR

ASSEMBLER REFERENCE (Cont.)

Pseudo Instruction

General:

ORG
END
EQU
SET
DS
DB
DW

Macros:

MACRO
ENDM
LOCAL
REPT
IRP
IRPC
EXITM

Relocation:

ASEG	NAME
DSEG	STKLN
CSEG	STACK
PUBLIC	MEMORY
EXTRN	

Conditional Assembly:

IF
ELSE
ENDIF

Constant Definition

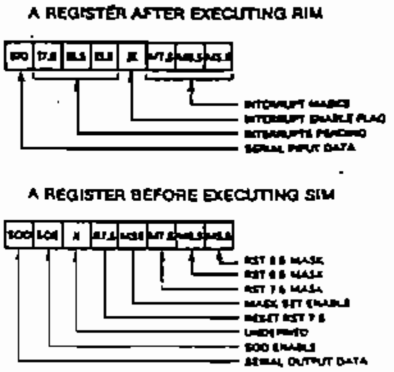
0BDH 1AH	Hex
105D 105	Decimal
72O 72Q	Octal
11011B 00110B	Binary
'TEST' 'A' 'B'	ASCII

RESTART TABLE

Name	Code	Restart Address
RST 0	C7	0000_{16}
RST 1	CF	0008_{16}
RST 2	D7	0010_{16}
RST 3	DF	0018_{16}
RST 4	E7	0020_{16}
TRAP	Hardware* Function	0024_{16}
RST 5	EF	0028_{16}
RST 5.5	Hardware* Function	$002C_{16}$
RST 6	F7	0030_{16}
RST 6.5	Hardware* Function	0034_{16}
RST 7	FF	0038_{16}
RST 7.5	Hardware* Function	$003C_{16}$

*NOTE: The hardware functions refer to the on-chip Interrupt feature of the 8085 only.

USE OF THE A REGISTER BY RIM AND SIM INSTRUCTIONS (8085 ONLY)


A REGISTER AFTER EXECUTING RIM

A REGISTER BEFORE EXECUTING SIM

Table 4. MCS-51™ Instruction Set Description

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Cyc
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	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 from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A w/Borrow	1	1
SUBB	A,#data	Subtract immed. data from A w/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 & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

LOGICAL OPERATIONS

Mnemonic		Destination	Byte	Cyc
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 A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

DATA TRANSFER

Mnemonic		Description	Byte	Cyc
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	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

DATA TRANSFER (cont.)

Mnemonic		Description	Byte	Cyc
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	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 with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order Digit ind. RAM w/A	1	1

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Cyc
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

Mnemonic		Description	Byte	Cyc
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 addr)	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 No Carry flag	2	2
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg. & Jump if Not Equal	3	2
CJNE	@Ri,#data,rel	Comp. immed. to ind. & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct,rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn —Working register R0–R7
direct —128 internal RAM locations, any I/O port, control or status register
@Ri —Indirect internal RAM location addressed by register R0 or R1
#data —8-bit constant included in instruction
#data16—16-bit constant included as bytes 2 & 3 of instruction
bit —128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr16 —Destination address for LCALL & LJMP may be anywhere within the 64-Kilobyte program memory address space.
addr11 —Destination address for ACALL & AJMP will be within the same 2-Kilobyte page of program memory as the first byte of the following instruction.
rel —SJMP and all conditional jumps include an 8-bit offset byte. Range is +127/-128 bytes relative to first byte of the following instruction.