BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Course: EEE 315 Microprocessor and Interfacing, Term: Jan 2016

Assignment on Modified Simple As Possible Computer (MSAP-2016)

Design Problem:

This assignment requires each student to submit a software design of an 8-bit computer which can execute instructions from an assigned instruction set (the list is attached below). Students are free to choose suitable simulation software (Circuit Maker or Proteus may be good choices for this design purpose). Each student has to implement 16 instructions assigned according to his/her student number.

Submission Procedure:

Students will submit their assignment in two phases. In the first phase, each student will submit a detailed block diagram of the computer architecture showing all the blocks (with appropriate control signals) necessary to perform the assigned instruction set. Along with the block diagram, a table showing active signals and microinstructions for every T state of fetch and execution cycles for each instruction is also to be submitted. Students may have to face a viva-voce to explain their architecture.

In the second phase of submission, students will submit their circuit level designs. They have to simulate and explain their designs on-site.

MSAP-2015 Specifications:

Designs to be submitted must conform to the following technical specifications:

1. 8-bit microcomputer.

2. Pipelining is recommended (but not necessary).

3. 64 kBytes of main memory (RAM) support.

4. A and B registers are accessible to programmers (Students may include additional temporary register if it is absolutely required).

5. 8-bit opcode for instructions.

6. Provision for hexadecimal input (students have to include a hexadecimal key encoder)

7. Provision for hexadecimal output (binary to hexadecimal converter is to be included)

8. Provision for run in single instruction mode (For example see SAP-1).

9. The computer has to wait for input. It will continue running when a switch is pressed after setting the input.

10. Flag register includes zero, sign, parity, carry and overflow flags.

11. Assembler program converts assembly code to machine code.

12. All ICs except timer, RAM and ROM are from 74XXX series.

13. Clock frequency 1kHz.

14. Provision for loading program from a ROM or from hexadecimal keypad to the system memory (RAM) at the beginning of the simulation.

15. List and number of all ICs used.

MSAP-2015 Instruction Sets

Student ID: XX1		Student ID: XX2		Student ID: XX3	
Opcode	**Instruction**	**Opcode**	**Instruction**	**Opcode**	**Instruction**
00	RCR A	00	JZ address	00	HLT
01	CALL address	01	NOT B	01	RET
02	MOV [address], A	02	CALL address	02	NOT A
03	JNZ address	03	RET	03	CALL address
04	MOV B, [address]	04	MOV B, byte	04	JA address
05	POP A	05	HLT	05	POP [address]
06	XCHG B, A	06	MOV A, [address]	06	MOV B, [Address]
07	HLT	07	PUSH A	07	MOV A, byte
08	PUSH [address]	08	OUT A	08	CMP B, [address]
09	RET	09	TEST A, [Address]	09	OUT B
0A	AND A, B	0A	IN B	0A	PUSH B
0B	OUT B	0B	POP [address]	0B	ADD A, B
0C	IN A	0C	SUB A, B	0C	IN [address]
0D	OR B, [address]	0D	XCHG [address], B	0D	SHL B
0E	CMP A, B	0E	RCL A	0E	SUB A, [address]
0F	ADD [address], B	0F	XOR [address], B	0F	XCHG [address], A

Student ID: XX4		Student ID: XX5		Student ID: XX6	
Opcode	**Instruction**	**Opcode**	**Instruction**	**Opcode**	**Instruction**
00	OUT B	00	PUSHF	00	XCHG [address], B
01	MOV A, [address]	01	IN B	01	NOT B
02	SHR B	02	RCL A	02	CMP
03	ADD [address], B	03	INC B	03	POP [address]
04	HLT	04	HLT	04	SHL B
05	DEC A	05	CALL address	05	HLT
06	NEG A	06	XCHG A, B	06	CALL address
07	CALL address	07	MOV B, byte	07	IN A
08	IN [address]	08	JLE address	08	MOV [address], A
09	PUSH B	09	RET	09	PUSH [address]
0A	OR A, B	0A	OUT A	0A	JA address
0B	POP A	0B	POPF	0B	MOV B, Byte
0C	XCHG B, [address]	0C	MOV [address], B	0C	NEG A
0D	RET	0D	XOR [address], A	0D	ADD B, byte
0E	TEST A, [address]	0E	SUB B, [address]	0E	OUT A
0F	JZ address	0F	CMP B, [address]	0F	RET

Student ID: XX7		Student ID: XX8		Student ID: XX9	
Opcode	**Instruction**	**Opcode**	**Instruction**	**Opcode**	**Instruction**
00	MOV A, byte	00	INC A	00	IN A
01	ADD A, [address]	01	OUT A	01	HLT
02	OUT B	02	CALL address	02	RET
03	IN [address]	03	SHR B	03	CMP A, B
04	HLT	04	SUB B, byte	04	INC [address]
05	RCR B	05	PUSH [address]	05	PUSH B
06	POP B	06	RET	06	XCHG [address], B
07	CALL address	07	TEST A, [address]	07	AND A, byte
08	XCHG [address], A	08	IN B	08	SHR A
09	RET	09	JE address	09	CALL address
0A	MOV B, A	0A	XCHG [address], B	0A	JAE address
0B	JBE address	0B	POP A	0B	OUT B
0C	CMP A, B	0C	MOV A, [address]	0C	POP [address]
0D	PUSH A	0D	HLT	0D	MOV B, byte
0E	XOR [address], B	0E	OR [address], A	0E	DEC [address]
0F	DEC A	0F	NOT B	0F	MOV [address], A

Student ID: XX0	
Opcode	**Instruction**
00	ADD [address], byte
01	SUB A, [address]
02	HLT
03	IN A
04	MOV B, A
05	POP B
06	XCHG [address], B
07	JGE address
08	NOT B
09	RET
0A	CALL address
0B	RCR A
0C	CMP B, [address]
0D	OUT A
0E	MOV [address], A
0F	PUSH A

Note:

- Byte means immediate 1 byte data e.g., MOV B, 5H will insert 05 H in B register.
- [address] means the content of e.g., MOV B, [5H] means "copy the content of RAM location 05H to B".
- CMP operation is similar to SUB operation except it does not store the result.
- TEMP operation is similar to AND operation except it does not store the result.

MSAP-2015 Instruction Details

Syntax	Meaning
MOV A, byte	Moves immediate byte to A
MOV B, byte	Moves immediate byte to B
MOV A, [address]	Moves content of address to A
MOV [address], A	Moves content of A to address
MOV B, [address]	Moves content of address to B
MOV [address], B	Moves content of B to address
ADD A, byte	Adds A with immediate byte and stores the result in A
ADD B, byte	Adds B with immediate byte and stores the result in B
ADD A, [address]	Adds the value of A with content of address and stores the result in A
ADD [address], A	Adds content of address with the value of A and stores the result in address
ADD B, [address]	Adds the value of B with content of address and stores the result in B
ADD [address], B	Adds content of address with the value of B and stores the result in address
SUB A, byte	Subtracts immediate byte from A and stores the result in A
SUB B, byte	Subtracts immediate byte from B and stores the result in B
SUB A, [address]	Subtracts content of address from the value of A and stores the result in A
SUB [address], A	Subtracts A from the content of address and stores the result in address
SUB B, [address]	Subtracts content of address from the value of B and stores the result in B
SUB [address], B	Subtracts B from the content of address and stores the result in address
XCHG A, B	Exchanges the contents of A and B
XCHG A, [address]	Exchanges the contents of A and address
XCHG [address], A	Exchanges the contents of A and address
XCHG B, [address]	Exchanges the contents of B and address
XCHG [address], B	Exchanges the contents of B and address
INC A	Increments the content of A by 1
INC B	Increments the content of B by 1
INC [address]	Increments the content of address by 1
DEC A	Decrements the content of A by 1
DEC B	Decrements the content of B by 1
DEC [address]	Decrements the content of address by 1
NEG A	Negate A (2's complement negation)
NEG B	Negate B (2's complement negation)
AND A, byte	ANDs A with immediate byte and stores the result in A
AND B, byte	ANDs B with immediate byte and stores the result in B
AND A, [address]	ANDs the value of A with content of address and stores the result in A
AND [address], A	ANDs content of address with the value of A and stores the result in address
AND B, [address]	ANDs the value of B with content of address and stores the result in B
AND [address], B	ANDs content of address with the value of B and stores the result in address
OR A, byte	ORs A with immediate byte and stores the result in A
OR B, byte	ORs B with immediate byte and stores the result in B
OR A, [address]	ORs the value of A with content of address and stores the result in A
OR [address], A	ORs content of address with the value of A and stores the result in address
OR B, [address]	ORs the value of B with content of address and stores the result in B
OR [address], B	ORs content of address with the value of B and stores the result in address
XOR A, byte	XORs A with immediate byte and stores the result in A
XOR B, byte	XORs B with immediate byte and stores the result in B
XOR A, [address]	XORs the value of A with content of address and stores the result in A
XOR [address], A	XORs content of address with the value of A and stores the result in address
XOR B, [address]	XORs the value of B with content of address and stores the result in B

XOR [address], B	XORs content of address with the value of B and stores the result in address
CMP A, byte	Compares A with immediate byte and updates flags
CMP B, byte	Compares B with immediate byte and updates flags
CMP A, [address]	Compares the value of A with content of address and updates flags
CMP [address], A	Compares content of address with the value of A and updates flags
CMP B, [address]	Compares the value of B with content of address and stores the result in B
CMP [address], B	Compares content of address with the value of B and updates flags
TEST A, byte	ANDs A with immediate byte and updates flags
TEST B, byte	ANDs B with immediate byte and updates flags
TEST A, [address]	ANDs the value of A with content of address and updates flags
TEST [address], A	ANDs content of address with the value of A and updates flags
TEST B, [address]	ANDs the value of B with content of address and updates flags
TEST [address], B	ANDs content of address with the value of B and updates flags
JMP address	Jumps to address: unconditional
JZ address	Jumps if result of previous operation was zero
JNZ address	Jumps if not zero
JG address	Jumps if greater: signed
JGE address	Jumps if greater or equal: signed
JL address	Jumps if less: signed
JLE address	Jumps if less or equal: signed
JE address	Jumps if equal
JA address	Jumps if above: unsigned
JAE address	Jumps if above or equal: unsigned
JB address	Jumps if below: unsigned
JBE address	Jumps if below or equal: unsigned
PUSH A	Pushes the content of A to stack
PUSH B	Pushes the content of B to stack
PUSH [address]	Pushes the content of address to stack
POP A	Pops from stack to A
POP B	Pops from stack to B
POP [address]	Pops from stack to the addressed location
PUSHF	Pushes the flag register to stack
POPF	Pops the flag register from stack
CALL address	Calls a procedure from address
RET	Return from procedure
IN A	Takes an input from the input port and puts it inside the A register
IN B	Takes an input and puts it inside the B register
IN [address]	Takes an input and puts it as the content of address
OUT A	Displays the content of A in output port
OUT B	Displays the content of B in output port
HLT	Halts the computer
RCL A	Rotate A with carry to left by 1
RCL B	Rotate B with carry to left by 1
RCR A	Rotate A with carry to right by 1
RCR B	Rotate B with carry to right by 1
SHL A	Shift A to left by 1
SHL B	Shift B to left by 1
SHR A	Shift A to right by 1
SHR B	Shift B to right by 1