Answer Sheet to Review Question

1**. What is the meaning of : SR<- Mem[MAR]?**
It's a memory -read operation. The contents of the address in MAR(memory-address-register) are fetched into the memory output SR(storage-register).

2. **Explain how a microprogramming computer executes ISA instructions.**
In a microprogrammed computer, the predetermined microinstruction which are usually stored in ROM, define the control point (activation of a particular data path) required throughout the system. Additionally the microinstruction controls the sequencing of operation interpretion. Usually the ISA instructions decode to an initial address to a microinstruction address. The microinstruction specifies the control point values as well as the expected address of next microinstruction .In this way, ISA instruction are run smoothly with microprogrammed decoders.

3**. Explain the difference between "vertical microcode", and "horizontal microcode".**
In some machines, the control points (the activation of a particular data path) are encoded to make more efficient use of microprogram storage This type of microinstruction is referred to as vertical. Vertical microinstructions usually save space by being encoded and make the microcode more compact and efficient, but need extra decode time and slow the processor time. On the other hand, other machines partition control point actions into many different classes, any of which can be activated within any given cycle. Usually each bit of microcode control value represents one control action. These are referred to as horizontal microinstructions. They run faster and use up more ROM space and make the microcode memory less efficient.

4**. The three broad classifications of computers used in this course --- L/S, R?M< and R+M differ in several essential features. State the properties of the three classes, give a "type machine" of reach class, and identify points on which they differ.**

L/S machine: (IBM RS/6000, IBM Power PC, MIPS, DEC Alpha, HP RISC)
All values must be loaded into registers before execution can take place, also called RISC architecture. Try to reduce the amount of complexity in the instruction set and addressing mode, regularize the instruction format to be consistent, so that regularity of executions is established and decoding, pipelining are made much easier. The number of registers are increased and fixed-sized instructions with uniform field interpretion and regular execution sequence are achieved.

R/M (IBM 36./370, Intel x86)
include instructions that can operate both on registers and with one of the operands residing in memory. The R/M architecture anticipated some variability in instructions size, addressing modes and field interpretion. They try to strike a balance between reducing complexity and representation flexibility., representation compactness and execution speed. The R/M address has some potential of pipelining.

R+M (VAX, M680X0)
allow formats to include operands from both registers and memory. R+M uses an extended set formats and registers modes to allow a powerful and varied specifications of operands and operations types.\, and achieve the goal of compact and flexible representation of program actions. Extremely difficult to design and developing pipelining is also quite difficult. Execution of each instruction is slowed down.

5. Explain the difference between 1's complement and 2's complement representations.

1's complement: -a=not a. e.g. -(0101)=1010
easy to implement, but with undesirable properties.
-(0)=1111<> 0
1's complement has the same numbers of positive and negative representations.

2's complement:
-a = 0-a, e.g., -(0101)=1011
little care is needed after conversion.
has one more negative representation than positive representation.

Actually 2's complement = 1's complement +1

6. Floating point representations vary widely. Express 31.25 in IBM S/390, VAXZ and IEEE standard floating point.

IBM s/390:

1	7	24
s	char	Mantissa

s=0 (positive)
char - bias =2   bias=64   so char = 66 = $(100010)_2$
mantissa = $2^{24} * 31.25 / 16^2$ = 0001111101000.....
So representation is: 01000010000111110100000000000000

VAX

1	8	23
s	char	Mantissa

s=0
char - bias =5   bias=128   so char = 133 = $(10000101)_2$
mantissa: $31.25*2^{23}/2^5$=111101000... (use left of hidden point rule)
So representation is: 01000010111110100000000000000000

IEEE

1	8	23
s	char	Mantissa

s=0
char - bias = 4   bias = 127   so char = 131 = $(10000011)_2$
mantissa = $31.25 / 16 * 2^{23}$ = 11110100...(use left of point rule)
So representation is : 01000001111110100000000000000000

7 **Write a "two operand and "three operand" assembly language program to compute the area of a triangle given by paris of single precisions floating point numbers that are the cartesian coordinates (1st quardrant of its vertices.**

suppose 3 points are (x1,¸ y1), (x2, y2),(x3,y3) respectively.
area = x1y2+ x2y3+x3y1-x2y1-x3y2-x1y3

suppose the 6 floating point numbers re stored in memory address x1,y1, x2,y2,x3,y3, respectively.

1).

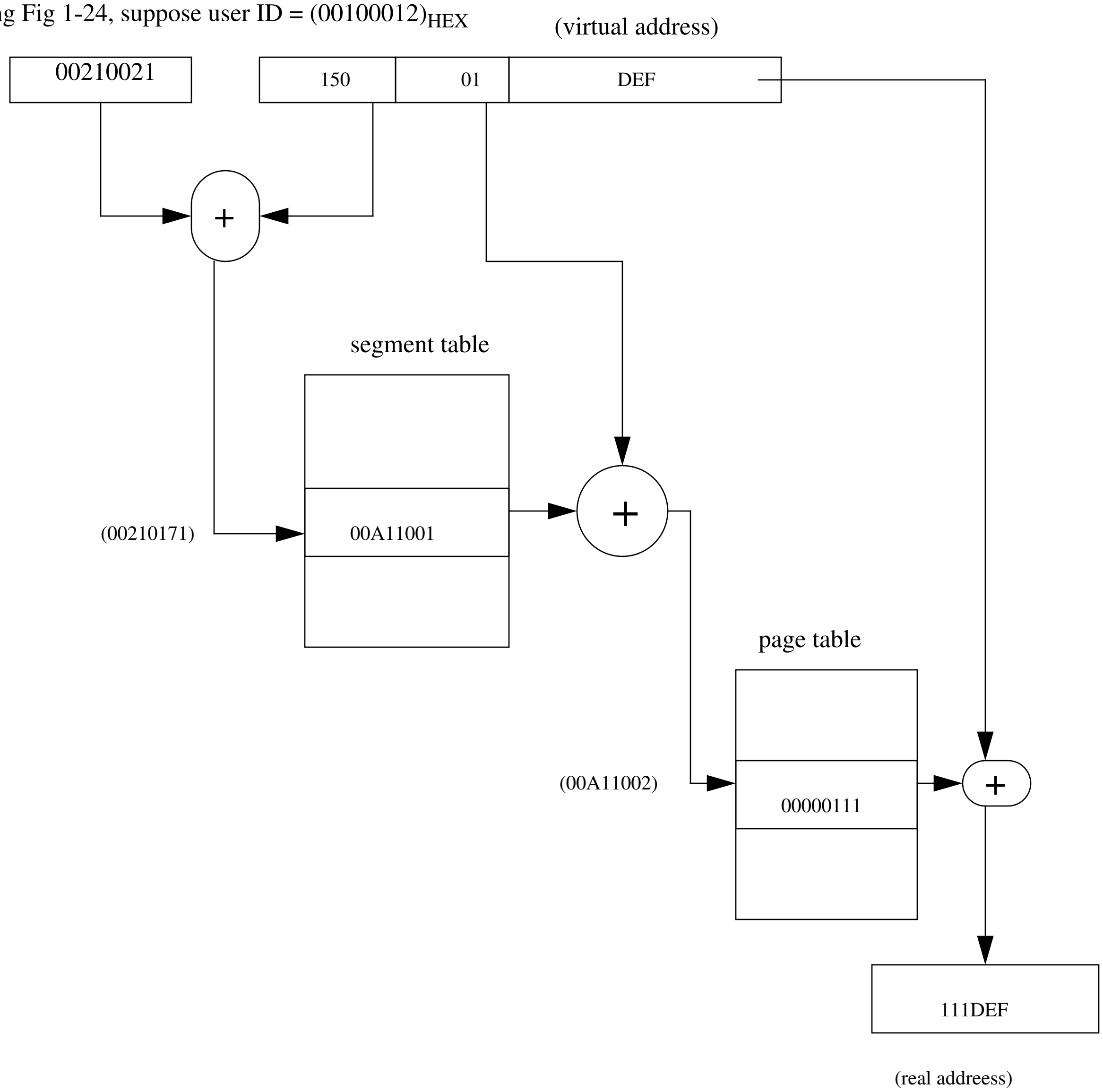
```
LD.F      R1,X1
LD.F      R2,Y2
LD.F      R3,XX
LD.F      R4,Y2
LD.F      R5,X3
LD.F      R6,Y3
MOVE.F    R7,R1
MOVE.F    R8,R2
MOVE.F    R9,R3
MOVE.F    R10,R4
MOVE.F    R11,R5
MOVE.F    R12,R6
MYP.F     R1,R4
MPY.F     R3,R6
MPY.F     R5,R2
MPY.F     R7,R10
MPY.F     R9,R12
MPY.F     R11,R8
ADD.F     R1,R3
ADD.F     R1,R5
SUB.F     R1,R7
SUB.F     R1,R9
SUB.F     R1,R11
ST.F      AREA, R1
```

2)

```
LD.F     R1,X1
LD.F     R2,Y1
LD.F     R3,X2
LD.F     R4,Y2
LD.F     R5,X3
LD.F     R6,Y3
MPY.F  R7,R1,R5
MPY.F  R8,R2,R6
MPY.F  R9,R3,R4
MPY.F  R10,R2,R4
MPY.F  R11,R3,R5
MPY.F  R12,R1,R6
ADD.F  R7,R8,R7
ADD.F  R7,R7,R9
SUB.F  R7,R7,R10
SUB.F  R7,R7,R11
SUB.F  R7,R7,R12
ST.F     AREA,R7
```

8. Set up a sample user ID, segment table and page table, and illustrate the conversion of a virtual address to a 24-bit physical address, using the simple mapping scheme in Figure 1.24

using Fig 1-24, suppose user ID = $(00100012)_{HEX}$



9. Illustrate timi~~ng for a 4~~ instruction sequence form one of your programs in Problem 7.

LD.F R1,X1

PC->TLB

CACHE MISS

MM->SR

SR->IR

DECODE

ADDRESS GENERATE

DATA FETCH
   CACHE MISS
   MM->SR
   EXECUTE R5<-SR
NEXT INSTR PREP
   (pc=pc+4)

MOVE.F R7,R1

PC->TLB

CACHE ->SR

SR->IR

DECODE

EXECUTE R1>-R7

NEXT INSTR PREP
   (pc=pc+2)