May 1, 1999

COMPUTER ENGINEERING DEPARTMENT

COE 308

COMPUTER ARCHITECTURE

Major Exam II

Second Semester (982)

Time: 7:30-9:30 PM

Student Name : _____

Student ID. : _____

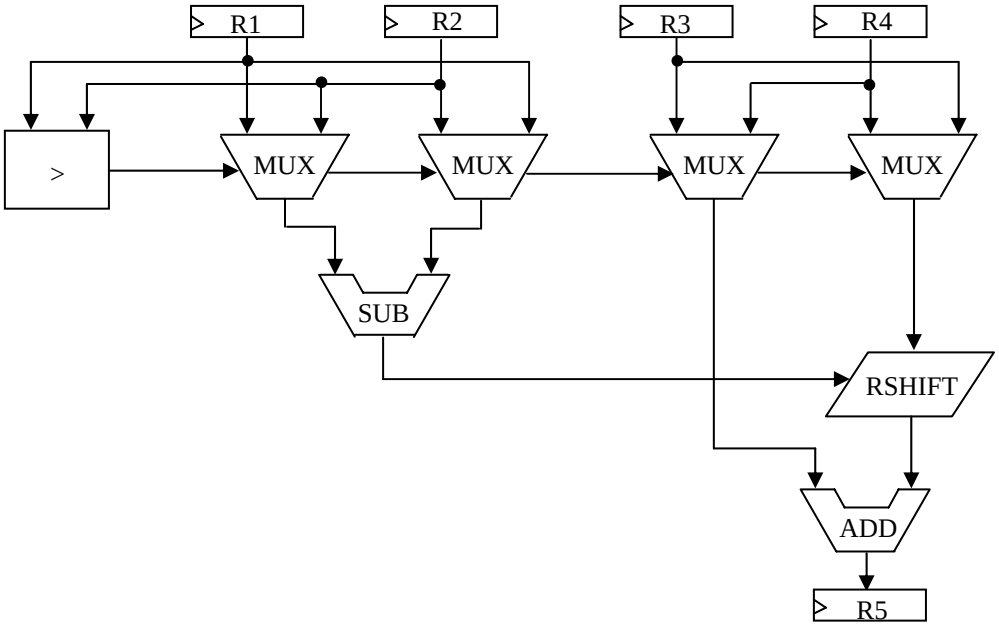
Question	Max Points	Score
Q1	20	
Q2	20	
Q3	10	
Q4	20	
Q5	15	
Q6	15	
Total	100	

Dr. Aiman El-Maleh

[20 Points]

Q.1. Suppose that it is required to design a hardware to execute the code given below, and you are given the implementation shown below as a possible solution. It is assumed that the propagation delay across the comparator (>) is 15ns, across the multiplexor (MUX) is 5 ns, across the right shifter (RSHIFT) is 15ns, and across each of the adder/subtractor blocks (ADD/SUB) is 20ns. Assume that in every clock cycle, the corresponding numbers in A[i] , B[I], C[I] and D[i] will be stored in R1, R2, R3, and R4, respectively. The result Z is assumed to be stored in R5.

```
for (i=0; i<15;i++){
  if  (A[i] > B[i]) {
       X= A[i] - B[i]
       Y= D[i] >> X;
       Z= C[i] + Y;
   } else {
      X= B[i] - A[i];
      Y= C[i] >> X;
      Z= D[i] + Y;
   }
 }
```



1. Determine the total time that will be taken for executing the code on the given implementation.

2. Modify the implementation to get a 4-stage pipeline such that the maximum clock period for each stage is 20ns. Then, determine the total time that will be taken for executing the code on the 4-stage pipeline implementation. What is the speedup factor compared to the original implementation.

3. Modify the implementation to get a 3-stage pipeline such that the maximum clock period for each stage is 25ns. Note here that the number of arithmetic blocks used in the modified implementation need not be the same. Then, determine the total time that will be taken for executing the code on the 3-stage pipeline implementation. What is the speedup factor compared to the original implementation.

[20 Points]

Q.2. A paging system has both a virtual memory and a real address data cache, with the following characteristics:

- 40-bit virtual address
- 32-bit real address
- 16Kbyte pages
- 2-way set associative TLB with 32 entries in total
- 4-way set associative data cache with 2048 lines in total

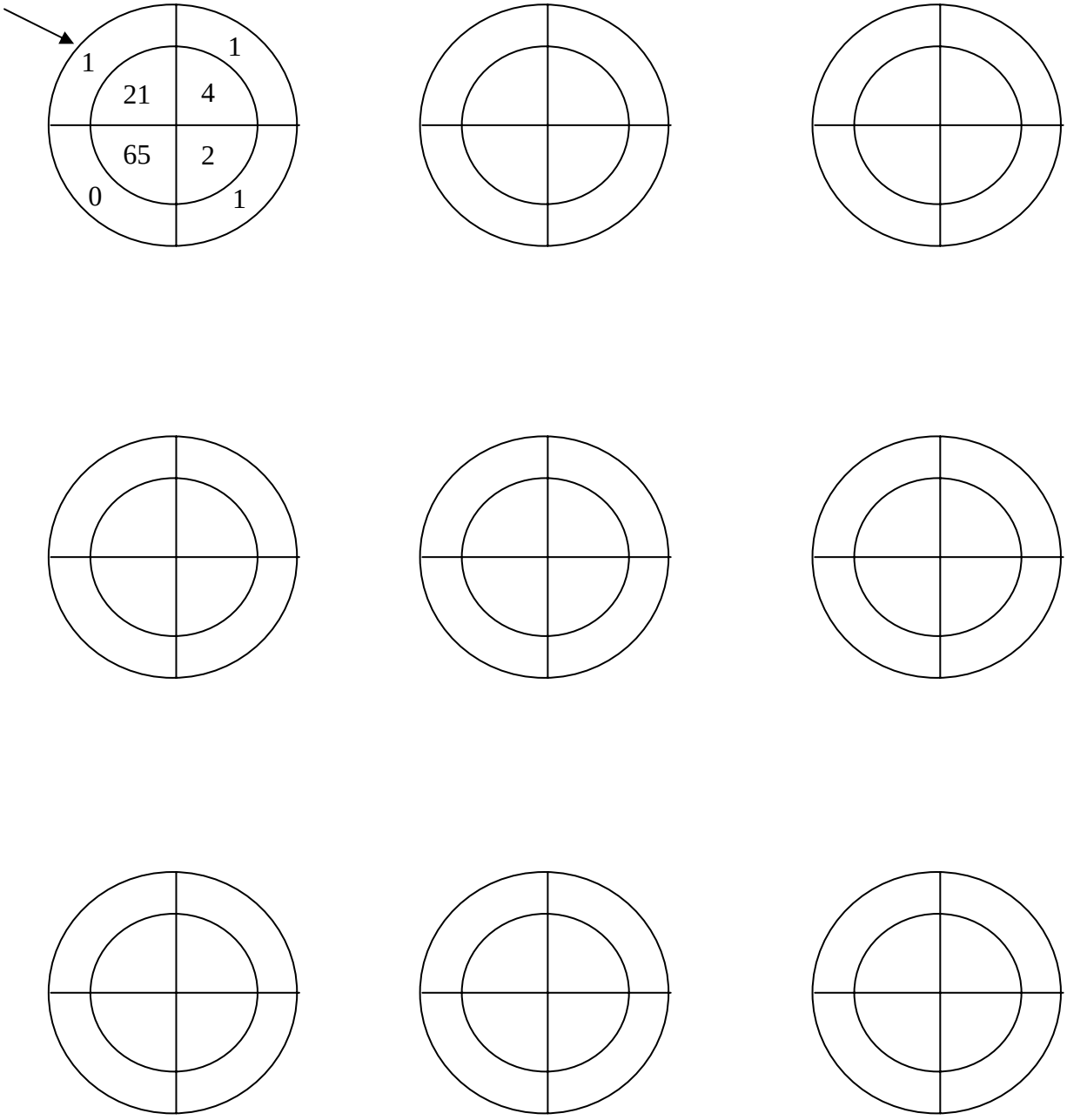
Draw the TLB and data cache, showing the division of bits in the virtual address, real address and the number of bits in the paths between the various component parts of the system. Each line contains one 32-bit word.

[10 Points]

Q.3. In a paged system, suppose that the main memory can only hold four pages at any instant and that the clock replacement algorithm is used for page replacement. Given that

the figure below indicates the current state of the clock replacement algorithm, list the pages in the main memory after each page is transferred assuming that the following pages are requested in the order shown:

33, 2, 65, 4, 43, 65, 2, 50



[20 Points]

Q.4. Assume that a 4-stage pipeline has the reservation table shown below:

Stages \ Time	0	1	2	3	4	5	6
1	X				X		
2		X	X				
3				X			
4						X	X

1. Determine the forbidden latency set and the initial collision vector.

2. Derive the state diagram for the collision vector and simplify it into a reduced state diagram.

3. List the simple cycles, and give the minimum average latency.

4. Is it possible to make an initiation every two cycles without collision? If not modify the reservation table, by introducing delays, to allow an initiation every two cycles.

[15 Points]

Q.5. A computer system has a five-stage instruction pipleline consisting of an instruction fetch unit (IF), an instruction decode unit (ID), an operand fetch unit (OF), an instruction

execution unit (IE), and an operand store unit (OS). Suppose that hazards due to true data dependency are recognized in the pipeline using valid bits. Suppose that the following code is to be executed through the pipleine:

```
1.   LD    R1, [100]    ;   load R1 by the content of address 100
2.   ADD   R1, R1, 1     ;   R1=R1+1
3.   LD    R2, [200]    ;   load R2 by the content of address 200
4.   ADD   R1, R1, R2   ;   R1=R1+R2
5.   SUB   R2, R1, R2   ;   R2=R1-R2
6.   ST    [100], R1    ;   store content of  R1 into address 100
7.   ST    [200], R2    ;   store content of  R2 into address 200
```

1. Identify potential data-dependency hazards between instructions in the code.

2. Draw a space-time diagram illustrating pipeline operation on the code, and determine the number of clock cycles needed to execute the code.

3. Assuming the existence of internal forwarding between the execution unit and the operand fetch unit, draw a space-time diagram illustrating pipeline operation on the code, and determine the number of clock cycles needed to execute the code.

[15 Points]

Q.6 A microprocessor has a 3-stage instruction pipeline, an instruction fetch unit (IF), an instruction decode unit (ID), and an instruction execution unit (IE). Assume that a conditional branch instruction outcome will be known in the instruction execution unit. Suppose that the following code is processed through the pipeline:

```
     1 MOV CX, 4        ;    CX=4
L1:  2 ADD AX, BX       ;    AX=AX+BX
     3 INC BX           ;    BX=BX+1
     4 DEC CX           ;    CX=CX-1
     5 JNZ L1           ;    Jump to L1 if not zero
     6 INC AX           ;    AX=AX+1
     MOV [100], AX      ;    Store content of AX into address 100
```

1. Draw a space-time diagram and compute the total code processing time assuming that the pipeline must be cleared after a branch instruction has been decoded regardless of its outcome during execution.

2. Draw a space-time diagram and compute the total code processing time assuming the use of a branch history table. Assume that when a branch instruction is executed initially it is not taken. Here it is assumed that the pipeline will be cleared only if the wrong execution path was taken.

3. Assume that there are two versions of the JNZ instruction, a delayed instruction by 2 cycles, called JNZD2, and a non-delayed JNZ instruction. Modify the code to take advantage of the delayed instruction and draw a space-time diagram and compute the total modified code processing time (Hint: the microprocessor has an instruction called **NOP** that performs no operation).