HIGH PERFORMANCE COMPUTER ARCHITECTURE 06-11-2015 **MATRICULATION NO.**_____

(MIDTERM TEST) **SURNAME**_____

FIRST NAME_____

1) (POINTS 30/40)
Consider the following snippet of code running on 4-ways out-of-order superscalar processor.
Initially, all registers contain zero.

```
lab1:   LW   R2,0(R1)
        MUL  R2,R2,R2
        SW   R2,0(R1)
        ADDI R1,R1,4
        BNE  R2,R0,lab1
```

Working hypothesis:
* the fetch, decode and commit stages are 4 instructions wide
* the instruction window has 3 slots
* we have 18 physical registers in the free pool
* the reorder buffer has unlimited size
* the integer multiplier has 4 stages
* the load/store queues have 3 slots each and a common effective-address calculation unit
* there are 4 ALUs for arithmetic and logic operations and for branching
* an ALU performs its operation in the same cycle when the operation is issued
* reads require 1 clock cycle (after the addressing phase)
* the register file has 4 input- and 4 output-ports
* there are 9 logical registers (including R0 which is hardwired to 0)
* the store operation leaves the issue stage as it is inserted in the store queue

In order to calculate the total cycles needed to execute 3 iterations of the above loop on such machine, complete the following chart until the end of the third iteration of the code fragment above, including the renamed stream the precise evolution of the free pool of the physical registers (the register map), the Instruction Window, the Reorder Buffer (ROB) and the Load Queue (LQ) and Store Queue (SQ).

Iter.	Instruction	F- Fetch (clock)	D - Decode (clock)	P - disPatch (clock)	I - Issue (clock)	X – eXecute (clock)	W – Write-back (clock)	C - Commit (clock)	Renamed Stream	Instruction Window Pi Pj Pk I Qj Qk	Reorder Buffer Ri Pi,old Cplt	Load Queue	Store Queue
1	**LW R2,0(R1)**	0							P2,0(P1)	P2 P1 - 0 0 0	R2 - -		
…	…												
…	…												

2) (POINTS 10/40) Given the same code:

```
mylab:  LW     R2, 0(R1)      ; read Xi
        MUL    R2, R2, R2     ; R2=R2*R2
        SW     R2, 0(R1)      ; write Xi
        ADDI   R1, R1, 4      ; update R1
        BNE    R1, R0, mylab  ; continue to loop if false
```

Show how this code can take advantage of software pipelining by statically reordering and renaming the instruction (write the actual code).

SOLUTION OF MIDTERM TEST 06-11-2015
(REVISED 27/10/2018)

1) In the following table,Cj,Ck are reported instead of Qj, Qk. Cj (Ck) represents the cycle when Qj (Qk) becomes zero. Similarly for the LQ/SQ Cj is reported instead of Qj.

```
==========================================================================================================================
PHYSICAL REGS:  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
                                        *  *
          qi:   1  1  1  1  1  1  1  1  0  0  1  1  1  1  1  1  1  1
          vi:  00 00 00 04 00 00 08 00 00 0C 00 00 00 00 00 00 00 00
==========================================================================================================================
REG.FILE: Ri:         1        2        3        4        5        6        7        8
          Pi:        10        9        -        -        -        -        -        -
          Qi:         0        0        0        0        0        0        0        0
          Vi:  00000008 00000000 00000000 00000000 00000000 00000000 00000000 00000000
==========================================================================================================================
STAGES:                F  D  P  I  X  W  C RENAMED-STR    INSTRUCTION-WINDOW                  REORDER-BUFFER          A  M  L  S  B  F  X
TOTAL SLOTS:           4  4  3  4 12  4  4 18            3                                   99                      4  1  1  0  1  4  1
BUSY SLOTS:            0  0  0  0  0  1  0 2             0                                   0                       0  0  0  0  0  0  0
STALLS:                0  4 11  9  0  0  6 0             0                                   0                       0  0  0  1  0  0  0
==========================================================================================================================
PC   INSTRUCTION       F  D  P  I  X  W  C Pi,Pj Pk Pl    IW#  OPCD Pi  Pj  Pk I/Pl  Cj Ck Cl  ROB# PC  Ri  oPi x s c  +-------------------+
000] LW   R2,0(R1)     0  1  2  3  4  6  7 P2,0(P1)      ----   LW P2  P1  -   0     2  -  -   ---- 000 R2  -   0 0 1  |LQ(0 )             |
001] MUL  R2,R2,R2     0  1  2  6  6 11 12 P3,P2,P2      ----  MUL P3  P2  P2  -     6  6  -   ---- 001 R2  P2  0 0 1  |PC   OP Pi  EFAD Ci|
002] SW   R2,0(R1)     0  1  2  4  5 11 12 ,0(P1)<-P3    ----   SW -   P3  P1  0     -  2  -   ---- 002 -   -   1 0 1  |---- LW P2  0000  6|
003] ADDI R1,R1,4      0  1  3  4  4  5 12 P4,P1,4       ---- ADDI P4  P1  -   4     3  -  -   ---- 003 R1  P1  0 0 1  |---- LW P5  0004  8|
004] BNE  R2,R0,-5     1  2  4  5  5  6 12 ,P3,P0,-5     ----  BNE -   P3  P0  -5    -  4  -   ---- 004 -   -   0 0 1  |---- LW P8  0008 11|
005] LW   R2,0(R1)     2  3  4  5  6  8 13 P5,0(P4)      ----   LW P5  P4  -   0     5  -  -   ---- 000 R2  P3  0 0 1  +-------------------+
006] MUL  R2,R2,R2     2  3  5  8  8 13 14 P6,P5,P5      ----  MUL P6  P5  P5  -     8  8  -   ---- 001 R2  P5  0 0 1
007] SW   R2,0(R1)     2  3  5  6  7 13 14 ,0(P4)<-P6    ----   SW -   P6  P4  0     -  5  -   ---- 002 -   -   1 0 1  +-------------------+
008] ADDI R1,R1,4      2  4  6  7  7  8 14 P7,P4,4       ---- ADDI P7  P4  -   4     6  -  -   ---- 003 R1  P4  0 0 1  |SQ(0 )             |
009] BNE  R2,R0,-5     3  4  6  7  7  8 14 ,P6,P0,-5     ----  BNE -   P6  P0  -5    -  6  -   ---- 004 -   -   0 0 1  |PC   OP Pi  EFAD Ci|
010] LW   R2,0(R1)     4  5  7  8  9 11 15 P8,0(P7)      ----   LW P8  P7  -   0     8  -  -   ---- 000 R2  P6  0 0 1  |---- SW P3  0000 11|
011] MUL  R2,R2,R2     4  5  7 11 11 16 17 P9,P8,P8      ----  MUL P9  P8  P8  -    11 11  -   ---- 001 R2  P8  0 0 1  |---- SW P6  0004 13|
012] SW   R2,0(R1)     4  6  8  9 10 16 17 ,0(P7)<-P9    ----   SW -   P9  P7  0     -  8  -   ---- 002 -   -   1 0 1  |---- SW P9  0008 16|
013] ADDI R1,R1,4      4  6  8  9  9 10 17 P10,P7,4      ---- ADDI P10 P7  -   4     8  -  -   ---- 003 R1  P7  0 0 1  +-------------------+
014] BNE  R2,R0,-5     5  7  9 10 10 11 17 ,P9,P0,-5     ----  BNE -   P9  P0  -5    -  9  -   ---- 004 -   -   0 0 1
```

Therefore 18 cycles are needed for this configuration.

2) Typical latencies among instructions are assumed.

Software Pipelining

```
MYLAB: LW    R2,0(R1)
       MUL   R3,R2,R2
       SW    R3,0(R1)
       ADDI  R1,R1,4
       BNE   R1,R0,MYLAB
```

3-times unrolling

```
MYLAB:
1   LW    R2,0(R1)
2   MUL   R3,R2,R2
3   SW    R3,0(R1)
4   LW    R2,4(R1)
5   MUL   R3,0R2,R2
6   SW    4(R1),R3
7   LW    R2,8(R1)
8   MUL   R3,R2,R2
9   SW    8(R1),R3
10  ADDI  R1,R1,12
11  BNE   R1,R0,MYLAB
```

```
       LW    R2,0(R1)
       MUL   R3,R2,R2
       LW    R2,4(R1)
       ADDI  R1,R1,4
MYLAB: SW    -4(R1),R3 ;  Store X[i]
       ADDI  R1,R1,4
       MUL   R3,R2,R2 ;   Add X[i-1]
       BNE   R1,R0,MYLAB
       LW    R2,0(R1);    Load X[i-2]
       SW    4(R1),R3
       MUL   R3,R2,R2
       SW    0(R1),R3
```