HIGH PERFORMANCE COMPUTER ARCHITECTURE 22-06-2015 **MATRICOLATION NO.**__________________

SURNAME____________________

FIRST NAME____________________

1) (POINTS 16/40) Consider the following snippet of code running on 4-ways out-of-order superscalar processor. Initially, R1=0x1000, R3=0x3000, R7=0x0003 and the other registers contain zero.

```
lab1:   LW   R2,0(R1)
        ADDI R2,R2,1
        MUL  R4,R2,R2
        SW   R4,0(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 8 slots
* we have 12 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).

2) (POINTS 6/40) Consider the following program, assuming that A and B are variables in memory initialized to 0, and R1, R2, and R3 are registers:

```
P1   P2   P3
A=1  R1=A R2=B
     B=1  R3=A
```

Executions of this program can be characterized by the values returned by the loads at the end of its execution, i.e., by the values of R1, R2, and R3 at the end of execution.
Using these register values, list the executions (if any) that are
(a1) *not* coherent;
(a2) *not* sequentially consistent;
(a3) *not* TSO;
(a4) *not* weakly ordered.

In each case give a justification for your answer.

3) (POINT 6/40) Answer the same questions as in (2) for the following program:

```
P1   P2
A=1  B=1
R1=B R2=A
```

4) (POINT 6/40) Answer the same questions as in (2) for the following program:

```
P1   P2
A=1  B=1
R1=A R3=B
R2=B R4=A
```

5) (POINT 6/40) Answer the same questions as in (2) for the following program:

```
P1   P2
A=1  B=1
C=1  C=2
R1=C R3=C
R2=B R4=A
```

HIGH PERFORMANCE COMPUTER ARCHITECTURE 22-06-2015 - SOLUTION

(REVISED 26/10/2018)

EXERCIZE 1

```
==============================================================================================================================================
PHYSICAL REGS:  1  2  3  4  5  6  7  8  9 10 11 12
                   *                          *  *
          qi:   1  0  1  1  1  1  1  1  1  1  0  0
          vi:  00 0C 01 01 04 00 01 01 08 00 01 01
==============================================================================================================================================
REG.FILE: Ri:         1        2        3        4        5        6        7        8
          Pi:         2       11        -       12        -        -        -        -
          Qi:         0        0        0        0        0        0        0        0
          Vi:  00001008 00000000 00003000 00000001 00000000 00000000 00000003 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  8  4 12  4  4 12             8                                    99                       4  1  1  0  1  4  1
BUSY SLOTS:             0  0  0  0  0  1  0 3              0                                    0                        0  0  0  0  0  0  0
STALLS:                 0  2  0 25  0  0  5 2              0                                    0                        0  0  0  2  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] ADDI R2,R2,1       0  1  2  6  6  7  8 P3,P2,1        ---- ADDI P3  P2  -   1    6  -  -   ---- 001 R2  P2  0 0 1  |PC   OP Pi  EFAD Ci|
002] MUL  R4,R2,R2      0  1  2  7  7 12 13 P4,P3,P3       ----  MUL P4  P3  P3  -    7  7  -   ---- 002 R4  -   0 0 1  |---- LW P2  1000  6|
003] SW   R4,0(R1)      0  1  2  4  5 12 13 ,P0(P1)<-P4    ----   SW -   P4  P1  0    -  2  -   ---- 003 -   -   1 0 1  |---- LW P6  1004  8|
004] ADDI R1,R1,4       1  2  3  4  4  5 13 P5,P1,4        ---- ADDI P5  P1  -   4    3  -  -   ---- 004 R1  P1  0 0 1  |---- LW P10 1008 10|
005] BNE  R2,R0,-6      1  2  3  4  4  5 13 ,P3,P0,-6      ----  BNE -   P3  P0  -6   -  3  -   ---- 005 -   -   0 0 1  +-------------------+
006] LW   R2,0(R1)      2  3  4  5  6  8 14 P6,0(P5)       ----   LW P6  P5  -   0    5  -  -   ---- 000 R2  P3  0 0 1
007] ADDI R2,R2,1       2  3  4  8  8  9 14 P7,P6,1        ---- ADDI P7  P6  -   1    8  -  -   ---- 001 R2  P6  0 0 1  +-------------------+
008] MUL  R4,R2,R2      2  3  4  9  9 14 15 P8,P7,P7       ----  MUL P8  P7  P7  -    9  9  -   ---- 002 R4  P4  0 0 1  |SQ(0 )             |
009] SW   R4,0(R1)      2  3  4  6  7 14 15 ,P0(P5)<-P8    ----   SW -   P8  P5  0    -  5  -   ---- 003 -   -   1 0 1  |PC   OP Pi  EFAD Cl|
010] ADDI R1,R1,4       3  4  5  6  6  7 15 P9,P5,4        ---- ADDI P9  P5  -   4    5  -  -   ---- 004 R1  P5  0 0 1  |---- SW P0  1000 12|
011] BNE  R2,R0,-6      3  4  5  7  7  8 15 ,P7,P0,-6      ----  BNE -   P7  P0  -6   -  5  -   ---- 005 -   -   0 0 1  |---- SW P0  1004 14|
012] LW   R2,0(R1)      4  5  6  7  8 10 16 P10,0(P9)      ----   LW P10 P9  -   0    7  -  -   ---- 000 R2  P7  0 0 1  |---- SW P0  1008 16|
013] ADDI R2,R2,1       4  5  6 10 10 11 16 P11,P10,1      ---- ADDI P11 P10 -   1   10  -  -   ---- 001 R2  P10 0 0 1  +-------------------+
014] MUL  R4,R2,R2      4  5  6 11 11 16 17 P12,P11,P11    ----  MUL P12 P11 P11 -   11 11  -   ---- 002 R4  P8  0 0 1
015] SW   R4,0(R1)      4  5  6  8  9 16 17 ,P0(P9)<-P12   ----   SW -   P12 P9  0    -  7  -   ---- 003 -   -   1 0 1
016] ADDI R1,R1,4       5  8  9 10 10 11 17 P2,P9,4        ---- ADDI P2  P9  -   4    9  -  -   ---- 004 R1  P9  0 0 1
017] BNE  R2,R0,-6      5  8  9 10 10 11 17 ,P11,P0,-6     ----  BNE -   P11 P0  -6   -  9  -   ---- 005 -   -   0 0 1
```

Therefore 18 cycles are needed for this configuration.

EXERCIZE 2

```
P1   P2    P3
A:=1 R1:=A R2:=B
     B:=1  R3:=A
```

-NOT coherent.

All possible outcomes are coherent because all accesses to each memory variable are made by different threads. So interleaving of accesses to the same memory variable is arbitrary and all possible outcomes are coherent.

-NOT sequentially consistent

The two stores (A:=1 and B:=1) are in different threads, thus they are not ordered. However P2 and P3 cannot observe the two stores in different orders. According to SC, if P2 reads A=1 and P3 reads B=1 then P3 cannot read A=0. So executions resulting in (R1,R2,R3)=(1,1,0) are not sequentially consistent. In this outcome the load-to-store order has been violated since B:=1 is allowed to perform while the preceding load of A is not globally performed.

-NOT TSO

TSO is more relaxed than sequential consistency primarily because of the relaxation of the store-toload order. However, the relaxation of the store-to-load order cannot affect this code, since the stores are not followed by loads in P1 and P2. Thus (1,1,0) is also not compliant with TSO.

-NOT weakly ordered

Since WO systems do not order regular loads and stores, all outcomes are possible.

EXERCIZE 3

Answer the same questions as in (2) for the following program:

```
P1      P2
A:=1    B:=1
R1:=B   R2:=A
```

This is the sequence in Dekker's algorithm.

-NOT coherent.

All possible outcomes are coherent because all accesses to each memory variable are made by different processors. So interleaving is arbitrary.

-NOT sequentially consistent.

This is well-known. All outcomes are SC except for (R1,R2)=(0,0). In this outcome the store-toload order is violated.

-NOT TSO

All outcomes are possible in TSO. All sequentially consistent outcomes are also TSO. The only invalid SC outcome would be (0,0), which is a valid one under TSO because of the store-to-load order relaxation.

-NOT weakly ordered

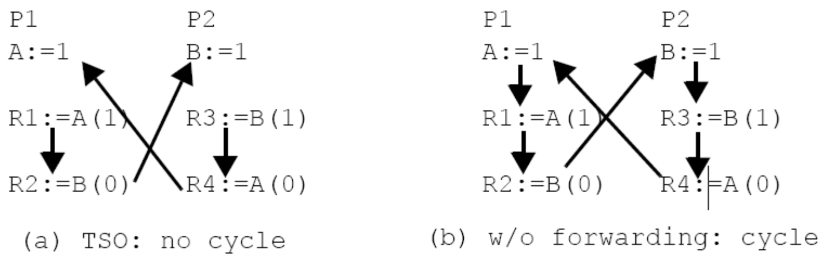
Since WO systems do not order regular loads and stores, all outcomes are possible.

EXERCIZE 4

Answer the same questions as in (2) for the following program:

```
P1     P2
A:=1   B:=1
R1:=A  R3:=B
R2:=B  R4:=A
```

To be at all correct, this sequence must return R1=1 and R3=1 (because of intra-process dependencies). So all outcomes in which R1 or R3 are 0 are incorrect under any model. Thus the only possible outcomes under ANY model are:
(R1,R2,R3,R4)=(1,0,1,0),(1,0,1,1),(1,1,1,0) or (1,1,1,1). In the following we restrict the discussion to these four outcomes.
-NOT coherent
All four outcomes are coherent. Considering accesses to A in isolation, the store to A is executed by P1, followed by the load of A (which must follow to respect intra-thread dependencies). The load of A in P2 can happen at any time, and thus can return either 0 or 1. The same outcome is true for B.
-NOT sequentially consistent
In SC, both R2 and R4 cannot be 0 at the same time. If P1 executes before P2, the outcome is (1,0,1,1). If P2 executes before P1, the outcome is (1,1,1,0). The outcome (1,1,1,1) is obtained when the two stores in P1 and P2 execute first and then all other accesses execute in any order after them. The only non-SC outcome is (1,0,1,0). Because R2 is 0, the code of P1 must precede P2's in any coherent order. Thus it is impossible for R4 to be 0.
Since all subsequent models are relaxed models, they must accept all SC outcomes. Therefore we focus on the one non-SC outcome only in the following, which is (1,0,1,0).
--NOT TSO
Is (1,0,1,0) a possible outcome of TSO? In TSO, the store-load order is not enforced. However load-load order must be enforced. Additionally, a load returning a value from the same thread's store is not ordered with the store, although it returns its value. Thus the two loads in each thread could be performed before the first store and (1,0,1,0) is possible. Therefore in TSO, all outcomes are possible. Note that, in a relaxed model that does not enforce store-load order and does not allow forwarding (such as IBM370), the stores and the loads in each thread would have to be in process order and (1,0,1,0) would be impossible. This is illustrated in the Figure below



--NOT W.O.
In WO, no order is imposed on regular loads and stores and thus all outcomes are possible.

EXERCIZE 5

Answer the same questions as in (2) for the following program:

```
P1      P2
A:=1    B:=1
C:=1    C:=2
R1:=C   R3:=C
R2:=B   R4:=A
```

This problem is more complex because up to 3 values can be returned by loads. R1 and R3 could potentially be 0, 1, or 2 and R2 or R4 could be 0 or 1. The total number of possibilities is 3x2x3x2=36. However, any outcome that returns R1=0 or R3=0 is incorrect: Because of intrathread dependencies, which are enforced in all cases, R1 and R3 must be equal to 1 or 2. So (0,x,x,x) and (x,x,0,x) are impossible outcomes in all cases. The only possible values for R1 and R3 in all cases are either 1 or 2. This leaves us with 2x2x2x2=16 possible outcomes for all models. We focus on those now.
--NOT coherent.
The only memory location that could cause coherence problems is C, because accesses to A and B are in different threads and so are not ordered by process order. The code restricted to accesses to C is:

```
P1      P2
C:=1    C:=2
R1:=C   R3:=C
```

If we look at all the possible orderings of the 4 accesses to C, the following must be enforced: if R1=2 then R3 must be 2 and if R3=1 then R1 must be 1. So (2,x,1,x) is not coherent.

Therefore, the impossible outcomes under coherence are (0,x,x,x), (x,x,0,x) and (2,x,1,x).
There are 3 possible values of (R1, R3) and 4 possible values of (R1,R2). Thus the number of coherent outcomes is 3x4 = 12 coherent outcomes.
-- NOT sequentially consistent.
SC must be at least coherent so (2,x,1,x) is not SC as well. If we look at the other accesses (on A and B), they are the same as in Dekker's algorithm. Thus we cannot have (R2,R4) =(0,0) and outcomes (x,0,x,0) are not SC.
Therefore the impossible outcomes under SC are (0,x,x,x), (x,x,0,x), (2,x,1,x) and (x,0,x,0).
Now we have only 3 possible values for (R1,R3) and 3 possible values for (R2,R4). The number of possible outcomes under SC is 3x3.
-- NOT TSO.
TSO relaxes the store-load order, including stores and loads to the same address. TSO is coherent (forwarding store buffer), so that (2,x,1,x) cannot be TSO. However, in TSO, the store of C and the load of C are not ordered. The load of C in P1 may return its value from the store buffer before the store to C has been globally performed. And the load of B in P1 may bypass the stores in the store buffer as well. Thus, (R2,R4) = (0,0) is a possible outcome under TSO, and the set of impossible outcomes under TSO are the same as under coherence.
Therefore, the impossible outcomes under TSO are (0,x,x,x), (x,x,0,x) and (2,x,1,x).
--NOT WO: Since WO systems do not order regular loads and stores, the only impossible outcomes are (0,x,x,x) and (x,x,0,x).