Alexandria University
Faculty of Engineering
Computers and Communications Department
Spring Final Exam, June 2017

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الحاسبات والاتصالات
امتحان نهاية الفصل الدراسي الثاني (يونيو ٢٠١٧)

Course Title and Code Number:
Advanced Computer Architecture (CC 423)
Time Allowed: 2 hours

اسم المقرر والرقم الكودي له:
معماريات الحاسب المتقدمة (CC 423)
الزمن: ساعتين

<u>Answer all the following questions:</u> (40 marks)

<u>Question 1:</u> (5 marks)

Consider enhancing a MIPS processor by adding vector hardware to it. When a computation is run in vector mode on the vector hardware, it is 10 times faster than the normal mode of execution. We call the percentage of time that could be spent using vector mode the percentage of vectorization.

a. Draw a graph that plots the speedup as a percentage of the computation performed in vector mode. Label the y-axis "Net speedup" and label the x-axis "Percent vectorization."
b. What percentage of vectorization is needed to achieve a speedup of 2?
c. What percentage of the computation run time is spent in vector mode if a speedup of 2 is achieved?
d. What percentage of vectorization is needed to achieve one-half the maximum speedup attainable from using vector mode?
e. Suppose you have measured the percentage of vectorization of the program to be 70%. The hardware design group estimates it can speed up the vector hardware even more with significant additional investment. You wonder whether the compiler crew could increase the percentage of vectorization, instead. What percentage of vectorization would the compiler team need to achieve in order to equal an addition 2x speedup in the vector unit (beyond the initial 10x)?

<u>Question 2:</u> (5 marks)

Examine the effect of the interconnection network topology on the *clock cycles per instruction (CPI)* of programs running on a 64-processor distributed-memory multiprocessor. The processor clock rate is 3.3 GHz and the base CPI of an application with all references hitting in the cache is 0.5. Assume that 0.2% of the instructions involve a remote communication reference. The cost of a remote communication reference is $(100 + 10h)$ ns, where h is the number of communication network hops that a remote reference has to make to the remote processor memory and back. Assume that all communication links are bidirectional.

a. Calculate the worst-case remote communication cost when the 64 processors are arranged as a ring, as an 8x8 processor grid, or as a hypercube. (Hint: The longest communication path on a 2^n hypercube has n links.)
b. Compare the base CPI of the application with no remote communication to the CPI achieved with each of the three topologies in part (a).
c. How much faster is the application with no remote communication compared to its performance with remote communication on each of the three topologies in part (a).

Question 3: (15 marks)

Consider a MIPS processor equipped with speculative out-of-order execution Tomasulo hardware. Study the processor performance when running the following code of the DAXPY operation, $Y=aX+Y$, for a vector length 100. Initially, R1 is set to the base address of array X and R2 is set to the base address of Y.

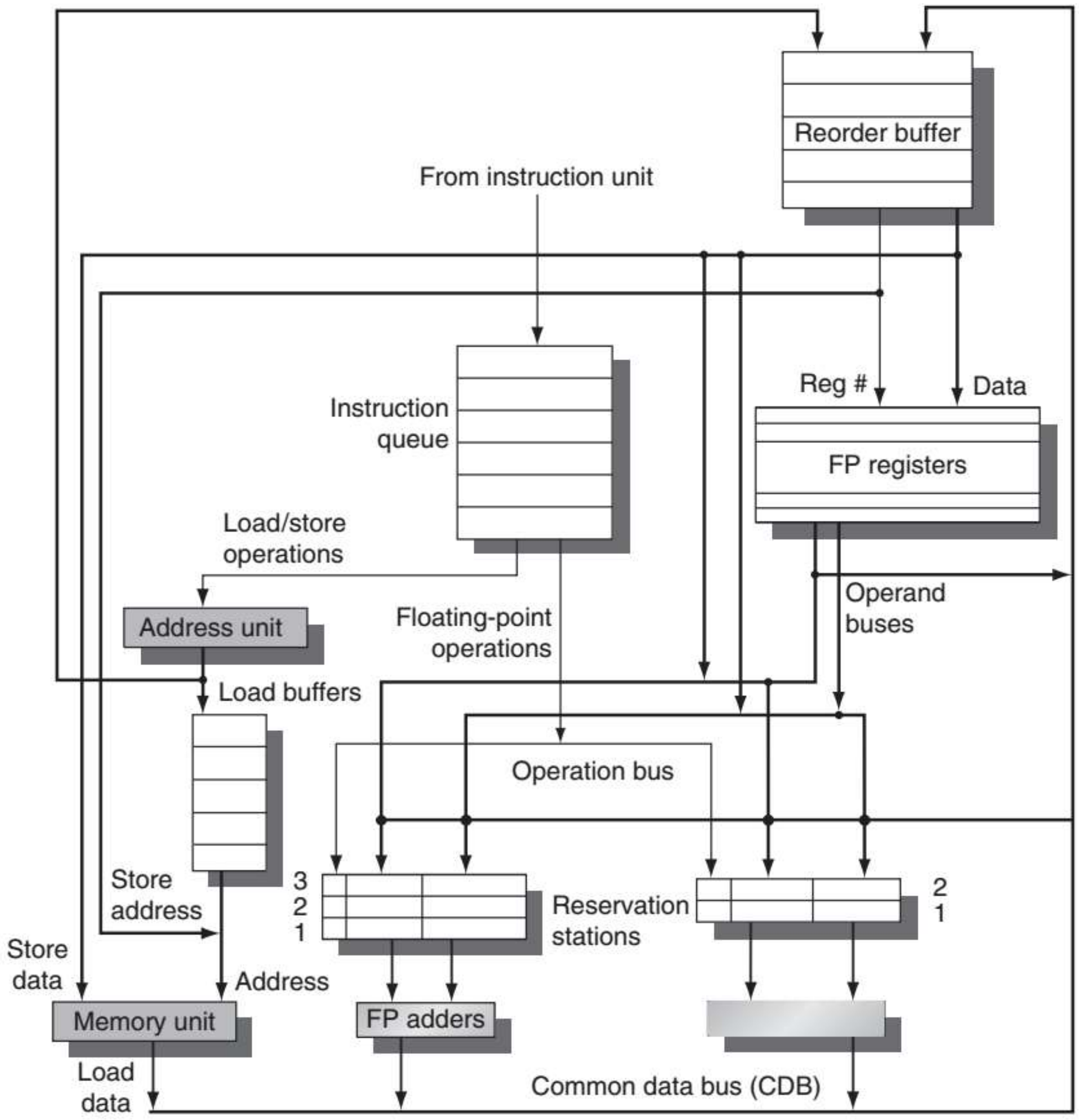
```
      DADDIU  R4,R1,#800   ; R1 = upper bound for X
foo:  L.D     F2,0(R1)     ; (F2) = X(i)
      MUL.D   F4,F2,F0     ; (F4) = a*X(i)
      L.D     F6,0(R2)     ; (F6) = Y(i)
      ADD.D   F6,F4,F6     ; (F6) = a*X(i) + Y(i)
      S.D     F6,0(R2)     ; Y(i) = a*X(i) + Y(i)
      DADDIU  R1,R1,#8     ; increment X index
      DADDIU  R2,R2,#8     ; increment Y index
      DSLTU   R3,R1,R4     ; test: continue loop?
      BNEZ    R3,foo       ; loop if needed
```

FU Type	Cycles in EX	Number of FUs	Number of reservation stations
Integer	1	1	5
FP adder	10	1	3
FP multiplier	15	1	2

Assume the following:

- Functional units are not pipelined.
- There is no forwarding between functional units; results are communicated by the common data bus (CDB).
- The execution stage (EX) does both the effective address calculation and the memory access for loads and stores. Thus, the pipeline is IF/ID/IS/EX/WB.
- Loads require one clock cycle.
- The issue (IS) and write-back (WB) result stages each require one clock cycle.
- There are five load buffer slots and five store buffer slots.
- Assume that the Branch on Not Equal to Zero (BNEZ) instruction require one clock cycle.

a. For this part use the single-issue speculative Tomasulo MIPS pipeline with the pipeline latencies from the table above. Show the number of stall cycles for each instruction and what clock cycle each instruction begins execution (i.e., enters its first EX cycle) for three iterations of the loop. How many cycles does each loop iteration take? Report your answer in the form of a table with the following column headers:
   - Iteration (loop iteration number)
   - Instruction
   - Issues (cycle when instruction issues)
   - Executes (cycle when instruction executes)
   - Memory access (cycle when memory is accessed)
   - Write CDB (cycle when result is written to the CDB)
   - Commit (cycle when the result is committed)
   - Comment (description of any event on which the instruction is waiting) Show three iterations of the loop in your table. You may ignore the first instruction.



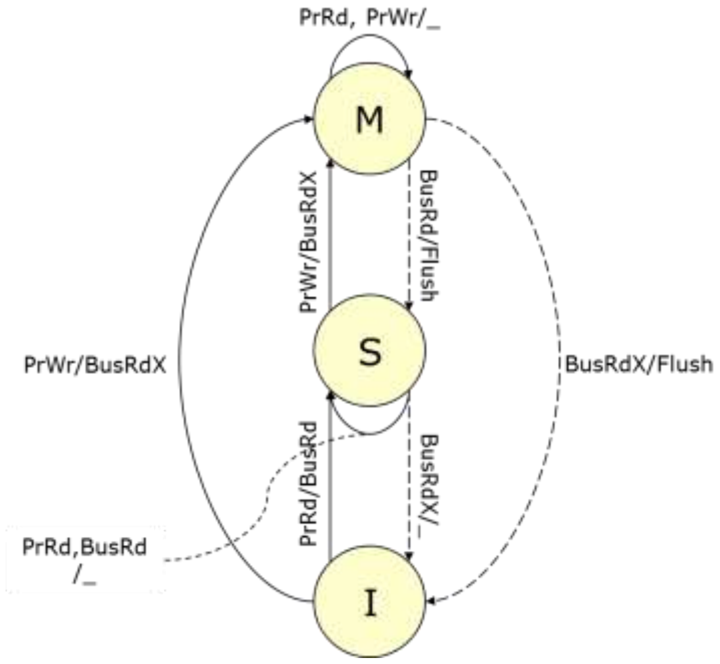
Speculative Tomasulo Hardware

b. Explain how RAW, WAW, and WAR hazards are resolved or treated in the basic Tomasulo's algorithm. In what way does a reorder buffer help speculation? What is the key points when introducing support for speculation? Give examples from (a) to support your answers.
c. Explain how dynamic branch prediction works when using a 2-bit prediction scheme. Use the program above to demonstrate how the prediction works. Also, describe how the prediction works for a program containing two or more branches?

Question 4: (15 marks)

The MSI protocol is a write-back write-invalidate cache coherence protocol where each block has Modified, Shared, and Invalid States as illustrated in the following Finite State Machine (FSM). The local processor events, bus transactions, and possible actions are defined in the following listing:

- ☐ Local processor events
  - ■ PrRd: read
  - ■ PrWr: write
- ☐ Bus transactions
  - ■ BusRd: read w/ no intent to modify
  - ■ BusRdX: read w/ intent to modify (read to own)
  - ■ BusWB: update memory
- ☐ Possible actions
  - ■ _: Nothing
  - ■ BusRd: send read request over the bus
  - ■ BusRdX: ownership (read to own) transaction
  - ■ Flush: copy modified block to memory



a) Lacking an Exclusive state causes unnecessary bus transactions. Modify the MSI protocol to incorporate a Clean Exclusive State E, indicate the changes needed to the cache system and draw the MESI protocol FSM.

b) MOESI is a five-state cache coherence protocol that avoids the need to write a dirty cache line back to main memory when another processor tries to read it. Instead, the Owned state allows a processor to supply the modified data directly to the other processor. A cache line in the owned state holds the most recent, correct copy of the data. Only one processor can hold the data in the owned state—all other processors must hold the data in the shared state. Now shared means shared and potentially dirty.
   Modify the MESI protocol to incorporate an Owned State O, indicate the changes needed to the cache system and draw the MOESI protocol FSM.

c) Spin locks are implemented using the following instructions:

```
            li    R2,#1
lockit:     exch  R2,0(R1)   ;atomic exchange
            bnez  R2,lockit  ;already locked?
```

   i. Write the assembly code to implement the atomic exchange operation: `exch R2,0(R1)` with load linked `ll` and store conditional `sc` instructions.

   ii. The problem with the above code is that the atomic exchange inside the loop includes a write which invalidates all other cached copies and generates considerable bus traffic. Rewrite the spin locks procedure code to resolve this problem by applying “test and test&set” exchange procedure.