Alexandria University
Faculty of Engineering
Computers and Communications Department
Mid-term Exam, April 2017

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الحاسبات والاتصالات
امتحان نصف الفصل الدراسي الثاني (إبريل ٢٠١٧)

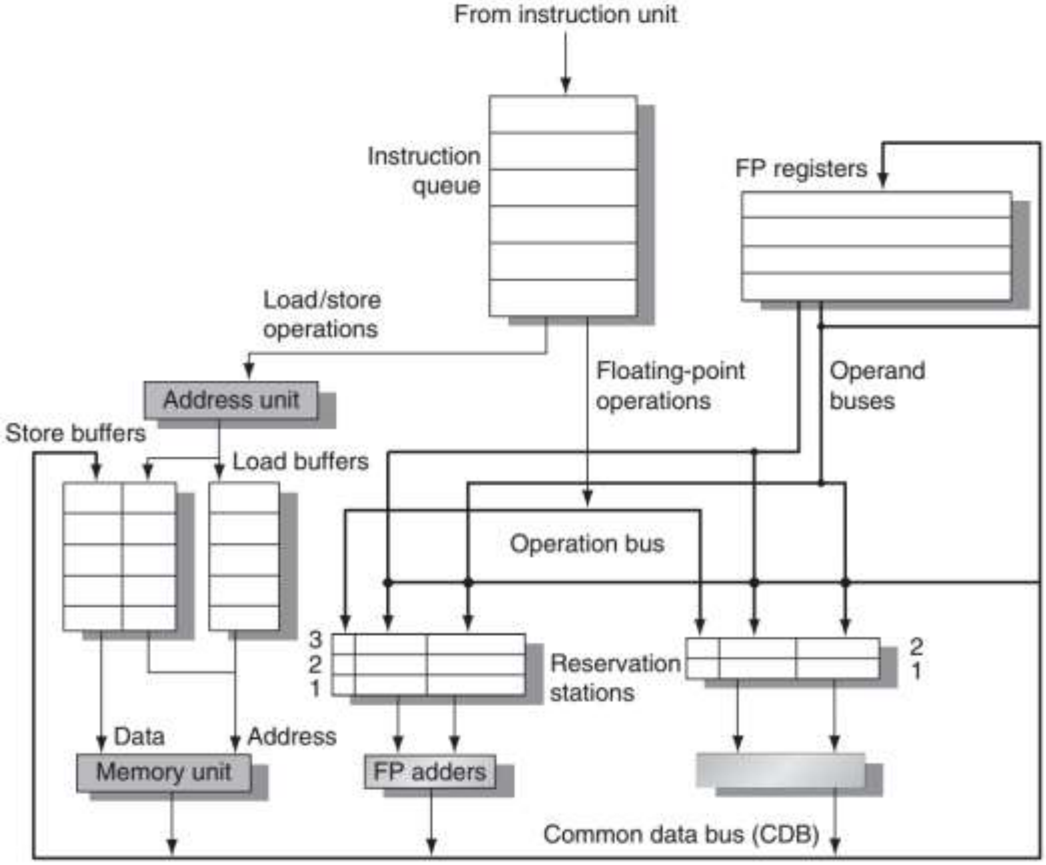
Course Title and Code Number:
Advanced Computer Architecture (CC 423)
Time Allowed: 1 hour

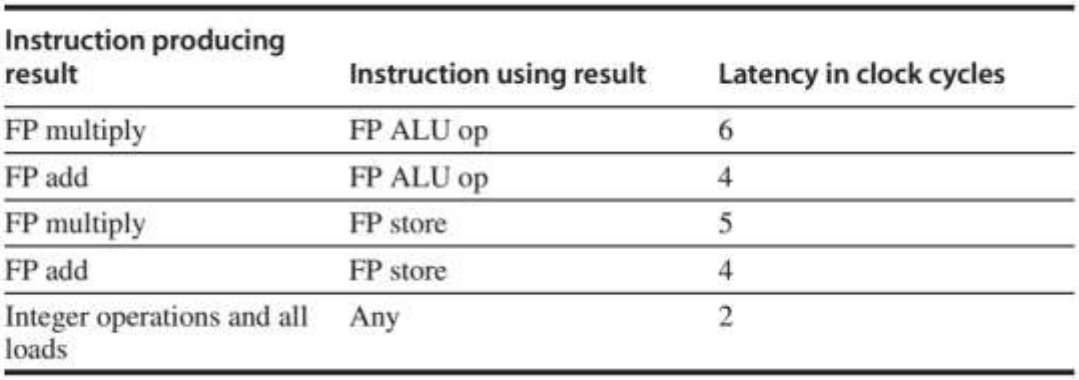
اسم المقرر والرقم الكودي له:
معماريات الحاسب المتقدمة (CC 423)
الزمن: ساعة واحدة

Answer all the following questions: (20 marks)

Question 1: (10 marks)

For this problem use the single-issue Tomasulo MIPS pipeline of Figure 1 with the pipeline functional units (FUs) latencies shown in Table 1.



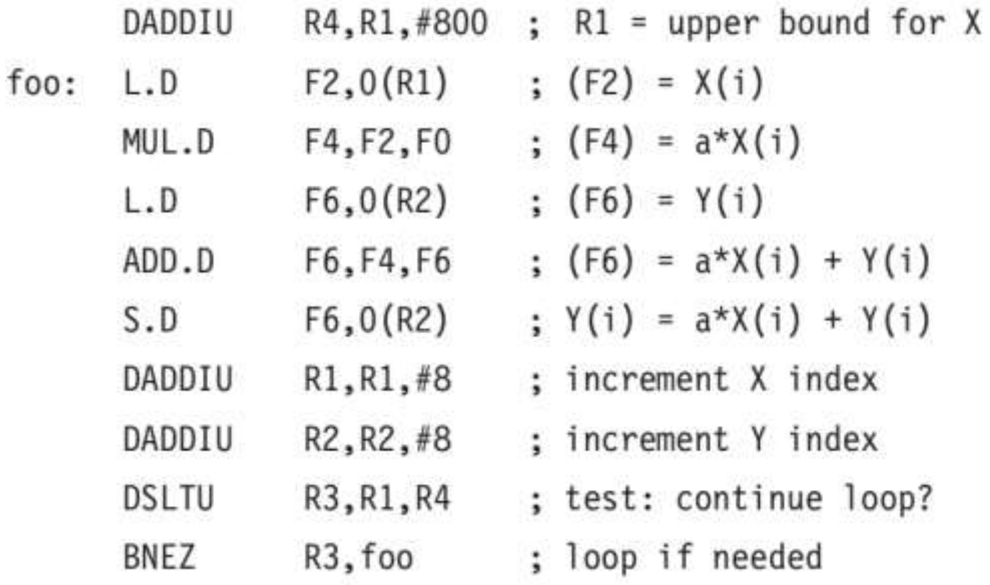


Instruction producing result	Instruction using result	Latency in clock cycles
FP multiply	FP ALU op	6
FP add	FP ALU op	4
FP multiply	FP store	5
FP add	FP store	4
Integer operations and all loads	Any	2

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.

The following loop is the so-called DAXPY loop (double-precision *aX* plus *Y*) and is the central operation in Gaussian elimination. The following code implements 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
```

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)
- 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.

Question 2: **(5 marks)**

When parallelizing an application, the ideal speedup is speeding up by the number of processors. This is limited by two things: percentage of the application that can be parallelized and the cost of communication. Amdahl's law takes into account the former but not the latter.

a) What is the speedup with N processors if 80% of the application is parallelizable, ignoring the cost of communication?

b) What is the speedup with 8 processors if, for every processor added, the communication overhead is 0.5% of the original execution time?

c) What is the speedup with 8 processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

d) What is the speedup with N processors if, for every time the number of processors is doubled, the communication overhead is increased by 0.5% of the original execution time?

e) Write the general equation that solves this question: What is the num-ber of processors with the highest speedup in an application in which P% of the original execution time is parallelizable, and, for every time the number of processors is doubled, the communication is increased by 0.5% of the original execution time?

Question 3: **(5 marks)**

Suppose we have a deeply pipelined processor, for which we implement a branch-target buffer for the conditional branches only. Assume that the misprediction penalty is always four cycles and the buffer miss penalty is always three cycles. Assume a 90% hit rate, 90% accuracy, and 15% branch frequency. How much faster is the processor with the branch-target buffer versus a processor that has a fixed two-cycle branch penalty? Assume a base clock cycle per instruction (CPI) without branch stalls of one.

Good Luck

Dr. Mohammed Morsy