CS 3853: Computer Architecture
Final
Fall 2014
Total Marks: 100
Time: 2 hour 30 min

Name:
Banner Id:
Alias:

1. [20 pt] Indicate **True** or **False** for the following statements:

   - Pipeline is a technique to improve throughput
   - The geometric mean of 0.1 and 0.4 is 0.002.
   - If system A is n% faster than system B, $ET_A/ET_B = 1 + n/100$.
   - In a single issue machine, CPI can be less than 1.
   - RAR is a type of data hazard.
   - Instruction in delay slot always gets executed no matter the outcome of the branch.
   - If branch target calculation and condition evaluation are done in the same stage, then *pipeline freeze* and *predict taken* strategies cause the same number of stalls.
   - Output dependence is a type of name dependence.
   - A statically scheduled processor can execute instructions out of order.
   - The key idea behind speculation is to allow instructions to execute and commit out of order
   - Main memory acts like a fully associate cache for disk.
   - Bigger cache blocks has no effect on conflict misses.
   - Cache miss is handled by hardware but page fault is handled by OS.
   - Locality that occurs in time is called temporal locality.
   - The higher the IPC the better it is
   - Multilevel cache is a way to reduce cache miss penalty.
   - The higher the associativity the smaller the tag is.
   - Prefetching can only be done in software.
   - Loop unrolling can degrade performance due to increased instruction cache miss.
   - Loop interchange is a compiler technique to improve locality of accesses.

2. [15 pt] CPU Performance and Amdahl's Law:

   (a) [2+3 pt] Suppose you can apply two enhancements - enh_1 and enh_2. enh_1 can be applied to 30% of the program whereas enh_2 can be applied to 20% of the first enhancement. Suppose the speed ups of the enhancements are 3.0 and 4.0 respectively.

      i. How much will be the overall speed up when only enh_1 is applied?

      ii. How much will be the overall speed up if you apply both enhancements together?

(b) [10 pt] Suppose we have the following instruction mix:

Instruction	Frequency (%)	CPI
Load	20	1
Store	15	2
Int ALU	50	1
Conditional Branch	10	2
Unconditional Branch	5	2

A new compiler is designed that replaces 30% of the original conditional branches with a new load followed by a new unconditional branch.

i. [2 pt] What is the original CPI?

ii. [2 pt] What are the frequencies of instructions with the new compiler?

iii. [2 pt] What is the new CPI using those frequencies?

iv. [2 pt] Assume that total instructions of a program with the original compiler is 100 millions. How many cycles do you need to execute the program?

v. [2 pt] How many total instructions will be there with the new compiler? How many cycles do you need to execute the program now?

3. [10 pt] Pipeline:
   Let us consider a multicycle MIPS pipeline shown in Figure 1.

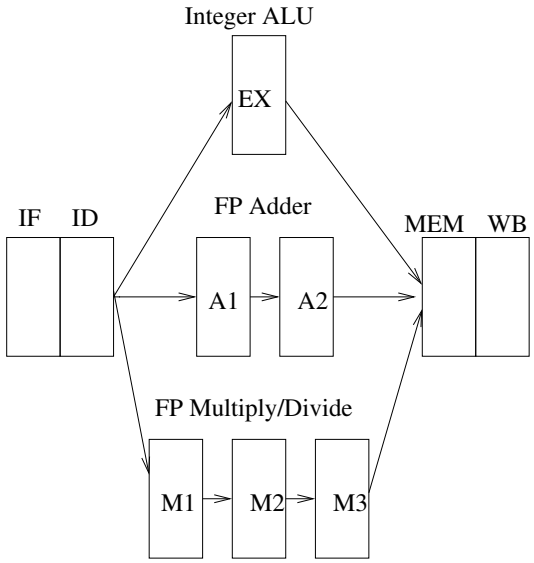

Figure 1: Pipeline stages.

- A branch target address is calculated in ID stage and condition is evaluated in EX stage. Branches are handled by using static predict taken policy.
- The pipeline forwards data from MEM or EX or M3 or A2 stage.
- Registers are written in the first half of a clock cycle and read in the second half.
- We have separate instruction and data memory.
- Branches are handled using **predict-not-taken** strategy.

(a) [6 pt] Assume that the branch is actually not taken. Fill up the following pipeline timing diagram.

Ins	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
LD R1, 0(R4)																		
BEQZ R1, L1																		
ADD.D F3, F1, F2																		
MUL.D F2, F3, F4																		
SD F3, 8(R2)																		
ADD.D F1, F3, F4																		

(b) [4 pt] Is it possible to have WAW and WAR hazards in this pipeline? If so, give an example of the corresponding hazards?

4. [20 pt] Instruction Level Parallelism:

   (a) [12 pt] Tomasulo's algorithm:
   We have the following assumptions.

   - Assume that you have 1 Integer, 1 FP Add/Sub, and 1 FP Divider/Multiplier Functional Unit. Integer FU takes 1 cycle, FP Add/Sub takes 2 cycles and FP Divider/Multiplier takes 5 cycles.
   - Assume **no hardware speculation** and **single issue**.
   - There is just **one CDB**.
   - Whenever there is a conflict for a functional unit or CDB, assume that the oldest (by program order) of the conflicting instruction gets access, while others are stalled.
   - Branches, loads, and stores use integer FU.
   - A branch instruction does not use CDB or memory.
   - Assume that the branch is not taken.
   - Loads and stores use the integer functional unit to perform effective address calculation during the EX stage. Memory access takes **1 cycle**. Data memory can serve only one access at a time.
   - If an instruction moves to CDB write stage in cycle **x**, then an instruction that is waiting on the same functional unit (due to a structural hazard) can start executing in cycle **x** unless the waiting instruction is dependent on the first one. In that case, the waiting instruction will start at cycle **x+1**.
   - Assume that the number of reservation stations is 2 for Integer, 1 for FP Divider/Multiplier and 1 for FP Add/Sub. Load, store and branch instructions use integer reservation stations.

   Complete Table 1 using Tomasulo's algorithm with the above specifications.

Instruction	Res. Station	IS	EX	Mem. Access	CDB Write
LD F2, 0(R1)					
MUL.D F4, F2, F0					
SD F4, 8(R1)					
MUL.D F4, F0, F1					
DSUBUI R1, R1, #32					
BEQZ R1, **L1**					
LD F1, 0(R1)					
ADD.D F3, F4, F2					

Table 1: Instruction sequence

(b) [4 pt] How does a brach target buffer work?

(c) [4 pt] What is the difference between statically scheduled and dynamically scheduled supersclar machine?

5. [35 pt] Memory hierarchy:

   (a) [20 pt] Let us assume a 2-way set associative 128 KB L1 cache with LRU replacement policy. The cache implements write back and no write allocate policy. Cache block size is 16 Byte. Page size is 64 KB. The system has a direct mapped TLB with16 entries. TLB also implements LRU policy. Virtual address is 24 bit and physical address is 20 bit long. The cache is virtually indexed and physically tagged.

      i. [2 pt] How many entries are there in the page table?

      ii. [2 pt] How many physical pages are there?

      iii. [2 pt] How many bits are used for TLB index and tag?

      iv. [2 pt] How many bits are used for cache index and tag?

v. [12 pt] Please fill up the following the table. All numbers are in hexadecimal.

Addr R/W	Binary Addr	TLB Tag	TLB Index	TLB Hit	Physical Addr	Cache Index	Cache Tag	Found Way
16103A W	000101100001000000111010							
17113C R	000101110001000100111100							
26112B R	001001100001000100101011							
16103B R	000101100001000000111011							
16103C W	000101100001000000111100							
23113D R	001000110001000100111101							
26103A R	001001100001000000111010							
17113D R	000101110001000100111101							
18113A R	000101110001000100111010							
23113C W	001000110001000100111100							
18113E R	000101110001000100111110							
23103A R	001000110001000000111010							

Table 2: Memory access sequence

Virtual Page	Physical Page
16	4
17	8
18	9
23	6
26	7

Table 3: Partial Page Table

TLB Index	TLB Tag	Physical Page	V
3			
6			
7			
8			

Table 4: TLB

(b) [3 pt] What is a victim cache?

(c) [6 pt] What are the different types of cache misses? How do you determine which miss is what?

(d) [3 pt] How does a write buffer work?

(e) [3 pt] What is the critical word first policy?