ECE4700J Computer Architecture

Summer 2022

HW #3

Due: **2:59am (Beijing Time) June 22nd, 2022**

*Please submit a single **PDF** file on Canvas*

Q1 (10%):

1) (5%) How many bypass paths are there in a 5-stage (F, D, X, M, W) pipeline that is 4 instructions wide? How many point-to-point wires and multiplexors (and what size muxes) does this bypass network requires, compared to a processor without bypassing?

2) (5%) What if you use k-clustering techniques such as group ALUs in 2 clusters? Repeat the above questions.

Q2 (20%):

1) (10%) Identify the write-read, write-write, and read-write dependencies in the instruction sequence below by entering each line pair with a dependency in the correct column of the table to the right. For example, if L1 and L4 had a write-write dependency (which they don't), you would enter L1-L4 in the column labeled "write-write"

```
L1: R1 = 100
L2: R1 = R2 + R4
L3: R2 = R4 - 25
L4: R4 = R1 + R3
L5: R1 = R1 + 30
```

write-read	write-write	read-write

2) (10%) Rename the registers from 1) to prevent dependency problems. You can assume there are free registers (R5 – R10). Show your work step by step.

Q3 (10%): Read about VLIW and list at least two limitations regarding VLIW. You can feel free to use internet for this question.

Q4 (40%): Given the following instruction sequence, and we also assume the following conditions:

a. The pipeline is a two-way superscalar.
b. The Memory stage takes two cycles to finish.
c. Loads have two-cycle load-to-use penalty (three cycle total execution latency).
d. Multiplication takes three cycles to finish.
e. All other instructions have single-cycle execution latency

Instruction	C0	C1	C2	C3	C4	C5	C6	….	
`ld [r1] → r3`									
`mult r3*r2 → r4`									
`add r2+r5 → r4`									
`or r4^r6 → r7`									
`sub r7-3 → r8`									
`ld [r9] → r7`									
…									

To enable the OoO execution, you can use a map table, ready table, reorder buffer and an issue queue. The following is the initial state.

Map Table

r1	p8
r2	p7
r3	p9
r4	p2
r5	p5
r6	p1
r7	p4
r8	p3
r9	p6

Ready Table

p1	Yes
p2	Yes
p3	Yes
p4	Yes
p5	Yes
p6	Yes
p7	Yes
p8	Yes
p9	Yes
p10	
p11	
p12	
p13	
p14	
p15	

p16	
p17	
p18	

Reorder Buffer

Instruction	To Free?	Done?
….		
…		

Issue Queue

Instruction	Src1	R?	Src2	R?	Dest	Bday

Question: How would this execution occur cycle-by-cycle? Show your work in details.

Q5 (20%):

1) (10%) Suppose that we want loads and stores to execute out-of-order with respect to each other. Under what circumstances in the code below can we execute instruction 5 before executing any others?

```
1  add x1, x1, x2
2  sw  x5, (x2)
3  lw  x6, (x8)
4  sw  x5, (x6)
5  lw  x8, (x3)
6  add x8, x8, x8
```

2) (10%) Under what circumstances can we execute instruction 4 in the code above before executing any others? Is it possible? Please explain your answer.

Optional Question:

OQ1 (10%): In an in-order single scalar 5-stage pipeline, consider a branch with the following behavior:

```
T, T, T, T, T, N, N, N, N, N, N, N, N, T, T, T, T, T, T, T, T, T,
T, N, N, N, N, N, T, T, T, T, T, T
```

N stands for not-taken and T stands for taken. You need consider only static, n-bit branch predictors as discussed in VE370, where m and n could be either 1 or 2. Your explanations can be based on inspection of the branch behavior; you do not have to work out actual branch misprediction rates in any case.

Q: Would you suggest using a 2-bit predictor for this branch? If yes, why? If not, which predictor should be used instead? Show your work.