Dipartimento di Elettronica e Informazione
**Politecnico di Milano**
20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (+39) 02-2399.3400
Fax (+39) 02-2399.3411

---

**Advanced Computer Architecture**

June 09, 2021

Prof. Marco Santambrogio

| Name |
| --- |
| Last Name |

Pick 2 out of th 3 problems, DO NOT DO ALL THE 3!

| | |
| --- | --- |
| Problem 1 (40%) | |
| Problem 2 (40%) | |
| Problem 3 (40%) | |
| Total (100%) | |

## Problem 1

Assume that the following code is executed on a CPU with SCOREBOARD and with the following units:

1 LOAD/STORE unit with Latency= 1
2 MULT units with latency = 10
1 DIVIDE unit with latency = 30
1 ADD/SUBD unit with latency 2

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| LD F6 32+ R2 | 1 | 2 | 3 | 4 |
| LD F2 45+ R3 | 5 | 6 | 7 | 8 |
| MULTD F0 F4 F2 | 6 | 9 | 19 | 20 |
| ADD F8 F2 F6 | 7 | 9 | 10 | 11 |
| DIVD F12 F8 F0 | 8 | 21 | 51 | 52 |
| SUBD F8 F6 F2 | 9 | 13 | 14 | 15 |

A. List all the possible conflicts in the code.

B. Is there a "configuration" that can respect the shown execution?
How many units? Which kind? What latency?

C. If the previous table was not correct, please, write the right one by having:
1 LOAD: 1cc
2 MULT: 10cc
1 DIVL 30cc
3 ADD/SUB: 1cc

## Problem 2

Consider the following access pattern on a two-processor system with a direct-mapped, write-back cache with one cache block and a two cache block memory. Assume the MESI protocol is used, with write-back caches, write-allocate, and invalidation of other caches on write (instead of updating the value in the other caches).

| Time | After Operation | P1 cache block state | P2 cache block state | Memory at block 0 up to date? | Memory at block 1 up to date? |
|---|---|---|---|---|---|
| 0 | P1: read block 0 | | | | |
| 1 | P2: write block 1 | | | | |
| 2 | P1: write block 0 | | | | |
| 3 | P2: read block 1 | | | | |
| 4 | P1: read block 1 | | | | |
| 5 | P2 : write block 1 | | | | |
| 6 | P1: read block 0 | | | | |
| 7 | P2: write block 1 | | | | |
| 8 | P1: read block 0 | | | | |
| 9 | P2: read block 1 | | | | |

## Problem 3

Design and describe (all the decisions have to be motivated) a 1-BHT and a 2-BHT able to execute the following assembly code.
(R0 is set to 10, R1 is set to 0)

```
LOOP:   LD      F3   0    R0
        ADDD    F1   F3   F3
        MULTD   F2   F3   F1
        ADDI    R1   R1   10
LOOP2:  LD      F3   0    R1
        MULTD   F2   F2   F3
        SUBI    R1   R1   1
        BNEZ    R1   LOOP2
        SUBI    R0   R0   10
        BNE     R0   R1   LOOP
```

The obtained result, in terms of miss predictions, is inline with theoretical characteristics of the two predictors? Please effectively support your answer.

## Answer

R0 is set to 10, R1 is set to 0
By entering LOOP, R1 is therefore = to 10
This implies that LOOP2 is iterated 10 times
After this first 10 iterations of LOOP2 R0 is = to 0, which means that LOOP is not iterated.
Within this context, we don't care about the possibility of having the same lane pointed by both the loops, because the worst initialization is the same.

1-BHT:
WORST case = LOOP: T LOOP2: NT
1st- LOOP2: miss
8 – LOOP2: correct
10th – LOOP2: miss
LOOP: miss
Miss: 27%

BEST case = LOOP: NT LOOP2: T
9 – LOOP2: correct
10th – LOOP2: miss
LOOP: correct
Miss: 9%

1

| | | | |
|---|---|---|---|
| LD | F6 | 32+ | R2 |
| LD | F2 | 45+ | R3 |
| MULTD | F0 | F4 | F2 |
| ADD | F8 | F2 | F6 |
| DIVD | F12 | F8 | F0 |
| SUBD | F8 | F6 | F2 |

RAW F6 11-14
RAW F6 11-16

RAW F2 12-13
RAW F2 12-14
RAW F2 12-16

RAW F0 13-15

RAW F8 14-15
WAR F8 15-16

WAW F8 14-16

**Answer 1.B**

NO: IT SUFFICES TO NOTICE THAT WAR ON F8 (15-16) IS NOT MANAGED

**Answer 1.C** ! NOTICE THAT ADD/SUBD HERE HAS A ≠ CC THAN BEFORE

| | Issue | Read Op | Exec Co. | Write R. |
|---|---|---|---|---|
| LD F6 32+ R2 | 1 | 2 | 3 | 4 |
| LD F2 45+ R3 | 5 | 6 | 7 | 8 |
| MULTD F0 F4 F2 | 6 | 9 | 9 | 20 |
| ADD F8 F2 F6 | 7 | 9 | 0 | 11 |
| DIVD F12 F8 F0 | 8 | 21 | 51 | 52 |
| SUBD F8 F6 F2 | 12 | 13 | 14 | 22 |

2

NO NEEDS TO COMMIT

| Time | After Operation | P1 cache block state | P2 cache block state | Memory at block 0 up to date? | Memory at block 1 up to date? |
|---|---|---|---|---|---|
| 0 | P1: read block 0 | E(0) | I | YES | YES |
| 1 | P2: write block 1 | E(0) | M(1) | YES | NO |
| 2 | P1: write block 0 | M(0) | M(1) | NO | NO |
| 3 | P2: read block 1 | M(0) | M(1) | NO | NO |
| 4 | P1: read block 1 | S(1) | S(1) | YES | YES |
| 5 | P2 : write block 1 | I | M(1) | YES | NO |
| 6 | P1: read block 0 | E(0) | M(1) | YES | NO |
| 7 | P2: write block 1 | E(0) | M(1) | YES | NO |
| 8 | P1: read block 0 | E(0) | M(1) | YES | NO |
| 9 | P2: read block 1 | E(0) | M(1) | YES | NO |

3

```
LOOP:   LD      F3   0    R0
        ADDD    F1   F3   F3
        MULTD   F2   F3   F1
        ADDI    R1   R1   10
LOOP2:  LD      F3   0    R1
        MULTD   F2   F2   F3
        SUBI    R1   R1   1
        BNEZ    R1   LOOP2
        SUBI    R0   R0   10
        BNE     R0   R1   LOOP
```

$R_0 = 10$ $R_1 = 0$

$R_1 = 10$ 10 ITERATIONS

$R_0 = 0$ $R_1 = 0$

$0 = 0$ $\Rightarrow$ 0 ITERATIONS

## 1-BHT COLLISION

```
LOOP:   LD      F3   0    R0
        ADDD    F1   F3   F3
        MULTD   F2   F3   F1
        ADDI    R1   R1   10
LOOP2:  LD      F3   0    R1
        MULTD   F2   F2   F3
        SUBI    R1   R1   1
T       BNEZ    R1   LOOP2
        SUBI    R0   R0   10
        BNE     R0   R1   LOOP
```

T->NT

$1 + 0 = 1$

## 2-BHT COLLISION

```
LOOP:   LD      F3   0    R0
        ADDD    F1   F3   F3
        MULTD   F2   F3   F1
        ADDI    R1   R1   10
LOOP2:  LD      F3   0    R1
        MULTD   F2   F2   F3
        SUBI    R1   R1   1
T = T   BNEZ    R1   LOOP2
        SUBI    R0   R0   10
        BNE     R0   R1   LOOP
```

$T_s \rightarrow T_w$ $T_w \rightarrow NT$

$1 + 1 = 2$

WE NOTICE THAT 1-BHT OUTPERFORMS 2-BHT, THAT IS NOT WHAT THEORETICALLY WE EXPECT

# EXTRA) DESCRIBE STATIC BRANCH PREDICTION TECHNIQUES

## BRANCH ALWAYS TAKEN

ALWAYS ASSUME, WHEN A BRANCH OCCUR, THAT IT IS TAKEN

PC = PC + target "MOVE WHERE BRANCH WANTS"

## BRANCH ALWAYS NOT TAKEN

ALWAYS ASSUME, WHEN A BRANCH OCCUR, THAT IT IS NOT TAKEN

PC = PC + 4 "MOVE TO NEXT INSTRUCTION"

## BACKWARD TAKEN FORWARD NOT TAKEN

ALWAYS ASSUME, WHEN A BRANCH OCCUR, THAT "WE ALWAYS ACCEPT TO MOVE BACK BUT REJECT FORWARD JUMPS"

## PROFILE-DRIVEN PREDICTION

DECISION BASED ON STATISTIC ANALYSIS OF PREVIOUS EXECUTIONS

## DELAY BRANCH PREDICTION

DELAY DECISION WAITING FOR VALUES TO BE AVAILABLE BY ADDING nop