**Appendix C2 - Sample Output for Project 2**

Please strictly follow the following format to generate the output report**, or your result will be deemed incorrect!**

*Format:*

cycle (cycle index in decimal representation)

$00: 0x(content in hexadecimal digits) # note that there is 1 space between “:” and “0x”

$01: 0x(content in hexadecimal digits)

…

…(other registers’ contents)

…

$31: 0x(content in hexadecimal digits)

PC: 0x(content in hexadecimal digits)

IF: 0x(bit stream fetched) [to\_be\_stalled/to\_be\_flushed]

ID: (mnemonic) [to\_be\_stalled/fwd\_EX-DM\_rs/t\_$x]

EX: (mnemonic) [fwd\_EX-DM/DM-WB\_rs/t\_$x]

DM: (mnemonic)

WB: (mnemonic)

PREDICTION: (“taken” or “not\_taken” or “n/a”)

CORRECTNESS: (“correct” or “incorrect” or “n/a”)

(2 lines here)

Note that

1. fwd\_EX-DM/DM-WB\_rs/t\_$x means “forwarding from EX-DM/DM-WB for rs/t $x”
2. Comments are here to help your understanding; they should not appear in the final output files.
3. If both source operands demand forwarding, then the one for rs goes first.
4. “PREDICTION: ” shows the result of your branch prediction and “CORRECTNESS: “ indicates whether it truly happened.
5. Put “n/a” after “PREDICTION: ” if the instruction in the IF stage is not a branch instructions as identified by the Branch Unit.
6. Put “n/a” after “CORRECTNESS: ” if the instruction in ID stage is not a branch instruction.
7. Please also output the contents of registers, fetched bit stream in the IF stage and the instruction types (mnemonics) in other pipeline stages **before** executing the instructions at each cycle.
8. Please report all detected hazards, i.e. stall, forwarding and flush, **right after** executing the instructions at each cycle.
9. It still verify the correctness of branch prediction when the branch instruction stall in ID stage, that is, evaluate the branch condition although the result of evaluation may be incorrect.

We list the following five examples to demonstrate the specified format.

*Example 1:*

Suppose that the iimage.bin indicates the following program:

lw $2, 0($3)

or $3, $1, $4

beq $2, $3, anyway

and $1, $1, $0

…

We first observe the instructions executed in the pipeline in the first few cycles:

IF ID EX DM WB

cycle 0 lw nop nop nop nop

cycle 1 or lw nop nop nop

cycle 2 beq or lw nop nop => make prediction!

cycle 3 and beq or lw nop => stall detected!

cycle 4 and beq nop or lw => forwarding detected!

The following is the content of snapshot.rpt after executing the above example.

(The content of registers are omitted)

*snapshot.rpt of Example 1:*

…

cycle 2

…(content of registers)

IF: 0x10520021

ID: OR

EX: LW

DM: NOP

WB: NOP

PREDICTION: not\_taken

CORRECTNESS: n/a

cycle 3

…(content of registers)

IF: 0x00200824 to\_be\_stalled

ID: BEQ to\_be\_stalled

EX: OR

DM: LW

WB: NOP

PREDICTION: n/a

CORRECTNESS: correct

cycle 4

…(content of registers)

IF: 0x00200824

ID: BEQ fwd\_EX-DM\_rt\_$3

EX: NOP

DM: OR

WB: LW

PREDICTION: n/a

CORRECTNESS: correct

…

*Example 2:*

Suppose that the iimage.bin indicates the following program:

lw $3, 0($2)

beq $2, $3, anyway

and $2, $5, $0

…

We first observe the instructions executed in the pipeline in the first few cycles:

IF ID EX DM WB

cycle 0 lw nop nop nop nop

cycle 1 beq lw nop nop nop => make prediction!

cycle 2 and beq lw nop nop => stall detected!

cycle 3 and beq nop lw nop => stall detected!

cycle 4 and beq nop nop lw

The following is the content of snapshot.rpt after executing the above example.

(The content of registers are omitted)

*snapshot.rpt of Example 2:*

…

cycle 1

…(content of registers)

IF: 0x10520021

ID: LW

EX: NOP

DM: NOP

WB: NOP

PREDICTION: not\_taken

CORRECTNESS: n/a

cycle 2

…(content of registers)

IF: 0x00A01024 to\_be\_stalled

ID: BEQ to\_be\_stalled

EX: LW

DM: NOP

WB: NOP

PREDICTION: n/a

CORRECTNESS: correct

cycle 3

…(content of registers)

IF: 0x00A01024 to\_be\_stalled

ID: BEQ to\_be\_stalled

EX: NOP

DM: LW

WB: NOP

PREDICTION: n/a

CORRECTNESS: correct

cycle 4

…(content of registers)

IF: 0x00A01024

ID: BEQ

EX: NOP

DM: NOP

WB: LW

PREDICTION: n/a

CORRECTNESS: correct

…

*Example 3:*

Suppose that the iimage.bin indicates the following program:

lw $3, 0($2)

or $1, $3, $4

and $2, $5, $0

xor $7, $8, $9

…

We first observe the instructions executed in the pipeline in the first few cycles:

IF ID EX DM WB

cycle 0 lw nop nop nop nop

cycle 1 or lw nop nop nop

cycle 2 and or lw nop nop => stall detected!

cycle 3 and or nop lw nop

cycle 4 xor and or nop lw => forwarding detected!

The following is the content of snapshot.rpt after executing the above example.

(The content of registers are omitted)

*snapshot.rpt of Example 3:*

…

cycle 2

…(content of registers)

IF: 0x00A01024 to\_be\_stalled

ID: OR to\_be\_stalled

EX: LW

DM: NOP

WB: NOP

PREDICTION: n/a

CORRECTNESS: n/a

cycle 3

…(content of registers)

IF: 0x00A01024

ID: OR

EX: NOP

DM: LW

WB: NOP

PREDICTION: n/a

CORRECTNESS: n/a

cycle 4

…(content of registers)

IF: 0x01093826

ID: AND

EX: OR fwd\_DM-WB\_rs\_$3

DM: NOP

WB: LW

PREDICTION: n/a

CORRECTNESS: n/a

…

*Example 4:*

Suppose that the iimage.bin indicates the following program:

addi $1, $2, 1

or $4, $2, $3

and $5, $1, $4

beq $1, $4, anywhere

sub $6, $7, $8

…

We first observe the instructions executed in the pipeline in the first few cycles:

IF ID EX DM WB

cycle 0 addi nop nop nop nop

cycle 1 or addi nop nop nop

cycle 2 and or addi nop nop

cycle 3 beq and or addi nop

cycle 4 sub beq and or addi => forwarding detected!

The following is the content of snapshot.rpt after executing the example, given above.

(The content of registers are omitted)

*snapshot.rpt of Example 4:*

…

cycle 3

…(content of registers)

IF: 0x10243020

ID: and

EX: OR

DM: ADDI

WB: NOP

PREDICTION: not\_taken

CORRECTNESS: n/a

cycle 4

…(content of registers)

IF: 0x00E83024

ID: BEQ fwd\_EX-DM\_rt\_$4

EX: AND fwd\_DM-WB\_rs\_$1 fwd\_EX-DM\_rt\_$4

DM: OR

WB: ADDI

PREDICTION: n/a

CORRECTNESS: correct

…

*Example 5:*

Suppose that the iimage.bin indicates the following program:

beq $0, $0, target

sub $6, $7, $8

target: and $2, $5, $0

…

We first observe the instructions executed in the pipeline in the first few cycles:

IF ID EX DM WB

cycle 0 beq nop nop nop nop

cycle 1 sub beq nop nop nop => flush detected!

cycle 2 and nop beq nop nop

The following is the content of snapshot.rpt after executing the above example.

(The content of registers are omitted)

*snapshot.rpt of Example 5:*

…

cycle 0

…(content of registers)

IF: 0x10000001

ID: NOP

EX: NOP

DM: NOP

WB: NOP

PREDICTION: not\_taken

CORRECTNESS: n/a

cycle 1

…(content of registers)

IF: 0x00E83024 to\_be\_flushed

ID: BEQ

EX: NOP

DM: NOP

WB: NOP

PREDICTION: n/a

CORRECTNESS: incorrect

cycle 2

…(content of registers)

IF: 0x00A01024

ID: NOP

EX: BEQ

DM: NOP

WB: NOP

PREDICTION: n/a

CORRECTNESS: n/a

…