10520 CS410001 - Computer Architecture 2017

Appendix C-2 - Sample Output for Project 2

Please refer to the specified output format to generate your output report. Since we are using a script to automatically check your results, **if the format does not match with our golden results, yours will be judged as incorrect and get no points**. You are encouraged to double check your results on the open test cases using the released golden simulator.

Format of each cycle:

```
cycle n                 # cycle number in decimal
$00: 0xhhhhhhhh         # list of register contents. There is a space between ":" and "0x"
$01: 0xhhhhhhhh         # 0xhhhhhhhh is the hexadecimal number of the register
…
…                       # other registers' contents
…
$31: 0xhhhhhhhh
$HI: 0xhhhhhhhh
$LO: 0xhhhhhhhh
PC: 0xhhhhhhhh
IF: 0xhhhhhhhh [to_be_stalled/to_be_flushed]     # show the content fetched
ID: instruction_d [to_be_stalled/fwd_EX-DM_rs/t_$x] # the decoded instruction name
EX: instruction_e [fwd_EX-DM_rs/t_$x]           # the instruction executing at EX
DM: instruction_m                  # the instruction executing at DM
WB: instruction_w                  # the instruction executing at WB
                                   # two empty lines followed here
                                   # you then start recording next cycle after this line
```

Notes:

1. fwd_EX-DM_rs/t_$x means "forwarding from EX-DM for rs/t $x"
2. Comments are here for your understanding; they should not appear in your output files. If you have any doubts or questions, please check with TA's.
3. If both source operands (r_s, r_t) demand forwarding, then process that of r_s first.
4. Report all occurring hazards, i.e. stall, forwarding and flush, **right after** executing the instructions at each cycle.

10520 CS410001 - Computer Architecture 2017

The following are 4 examples to help you get familiar with the format specification.

Example 1:

Suppose that the iimage.bin describes the following code:

```
lw $2, 0($3)
or $3, $1, $4
beq $2, $3, branch_line
and $1, $1, $0
…
```

We first observe the instructions executed in the pipeline for 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	
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 3
…                                   # content of registers
IF: 0x00200824 to_be_stalled
ID: BEQ to_be_stalled
EX: OR
DM: LW
WB: NOP

cycle 4
…                                   # content of registers
IF: 0x00200824
ID: BEQ fwd_EX-DM_rt_$3
EX: NOP
DM: OR
WB: LW
…
```

Example 2:

Suppose that the iimage.bin describes the following code:

```
lw $3, 0($2)
bne $1, $3, branch_code
and $2, $5, $0
…
```

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

	IF	ID	EX	DM	WB	
cycle 0	lw	nop	nop	nop	nop	
cycle 1	bne	lw	nop	nop	nop	
cycle 2	and	bne	lw	nop	nop	# stall detected!
cycle 3	and	bne	nop	lw	nop	# stall detected!
cycle 4	and	bne	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 2
…                                   # content of registers
IF: 0x00A01024 to_be_stalled
ID: BNE to_be_stalled
EX: LW
DM: NOP
WB: NOP

cycle 3
…                                   # content of registers
IF: 0x00A01024 to_be_stalled
ID: BNE to_be_stalled
EX: NOP
DM: LW
WB: NOP

cycle 4
…                                   # content of registers
IF: 0x00A01024
ID: BNE
```

EX: NOP
DM: NOP
WB: LW
…

Example 3:

Suppose that the iimage.bin describes the following code:

```
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 for 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

cycle 3
…                                   # content of registers
IF: 0x00A01024
ID: OR
EX: NOP
```

DM: LW
WB: NOP

cycle 4
… # content of registers
IF: 0x01093826
ID: AND
EX: OR fwd_DM-WB_rs_$3
DM: NOP
WB: LW
…

Example 4:

Suppose that the iimage.bin describes the following code:

```
addi $1, $0, 0x0000000D
mult $1, $1
mflo $1
mult $1, $0
…
```

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

	IF	ID	EX	DM	WB	
cycle 0	addi	nop	nop	nop	nop	
cycle 1	mult	addi	nop	nop	nop	
cycle 2	mflo	mult	addi	nop	nop	
cycle 3	mult	mflo	mult	addi	nop	# forwarding detected!
cycle 4	halt	mult	mflo	mult	addi	

snapshot.rpt of Example 4:

…
cycle 2
… # content of registers
IF: 0x00002012
ID: MULT
EX: ADDI
DM: NOP
WB: NOP

cycle 3

… # content of registers
IF: 0x00200018
ID: MFLO
EX: MULT fwd_EX-DM_rs_$1 fwd_EX-DM_rt_$1
DM: ADDI
WB: NOP

cycle 4
… # content of registers
IF: 0xFFFFFFFF
ID: MULT
EX: MFLO
DM: MULT
WB: ADDI
…