**CPRE 381- Intro to Computer Organization & Implementation**

**HW7**

**Due Date: April 14, 2017**

Name: Ninyuan Zhang

Section: A

Date: 4/13/2017

ID: 708235564

1. Problem 4.14.2; Patterson-Hennessy text-5th edition, Page 365. [5 points]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| instruction | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| lw r2,0(r1) | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |  |  |  |
| label1: beq r2, r0, label2 |  | IF | ID | ID | EXE | MEM | WB |  |  |  |  |  |  |  |
| lw r3,0(r2) |  |  | IF | IF | ID | EXE | MEM | WB |  |  |  |  |  |  |
| label2: sw r1,0(r2) |  |  |  |  | IF | nop | nop | nop | nop |  |  |  |  |  |
| beq r3,r0,label1 |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |  |
| add r1,r3,r1 |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |  |
| label1: beq r2,r0,label2 |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |  |
| lw r3,0(r2) |  |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |  |
| label2: sw r1,0(r2) |  |  |  |  |  |  |  |  |  | IF | ID | EXE | MEM | WB |

2. Problem 4.14.6; Patterson-Hennessy text-5th edition, Page 366. [5 points]

Need forwardings from EX/MEM to IF/ID and MEM/WB to IF/ID. If the branch depends on R-Type instruction, either one of them can be used. If it depends on lw instruction, only MEM/WB to IF/ID forwarding can be used.

3. Problem 4.16.2; Patterson-Hennessy text-5th edition, Page 367. [5 points]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| prediction | 0-NT | 01-NT | 00-NT | 01 -NT | 10-T |
| Actual Branch outcome | T | NT | T | T | NT |
| Riaht(1 WVrona(Q) Pred. | 0 | 1 | 0 | 0 | 0 |
| New Prediction | 1-NT | 00-NT | 01-NT | 10-T | 01-NT |

only the third T can be predictly correct, whitch is 20%

4. Problem 4.17.1; Patterson-Hennessy text-5th edition, Page 367. [5 points]

|  |  |
| --- | --- |
| instruction 1 | instruction 2 |
| invalid target address (ID) | overflow (EX) |

5. Problem 4.17.3; Patterson-Hennessy text-5th edition, Page 368. [5 points]

After the first exception occurs, the processor will save the address of the offending instruction in the EPC or the exception program counter, and transfer control to the operation system, OS, at some specified address.

The OS may take appropriate action, which may involve providing some service to the user program, either acting in response to overflow or stopping execution of the program or reporting an error.

After performing the appropriate action, the OS can terminate the program or continue execution, using the EPC to determine where to restart execution.

BNE stalled IF ID (bubble/stall) (bubble/stall) (bubble/stall)

LW IF ID EX MEM WB

6. Problem A.3; Patterson-Hennessy text-5th edition, Page A-82. [5 points]

No. It is never safe.

7. Problem A.5; Patterson-Hennessy text-5th edition, Page A-82. [5 points]

.ktext 0x80000080

sw $a0, save0

sw $a1, save1

mfc0 $k0, $13 # Move Cause into $k0

mfc0 $k1, $14 # Move EPC into $k1

addiu $v0, $zero, 0x44

slt $v0, $v0, $k0 # Ignore interrupts

bgtz $v0, \_restore

mov $a0, $k0 # Move Cause into $a0

mov $a1, $k1 # EPC into $a1

jal print\_excp # Print exception error msg

\_restore: lw $a0, save0

lw $a1, save1

lw $k0, -4($k1) # $k0 = previous instruction

srl $k0, $k0, 26 # $k0 = opcode of prev instr

ori $k1, $zero, 2 # opcode of j

beq $k0, $k1, \_delayslot #

ori $k0, $zero, 4 # opcode of beq

beq $k0, $k1, \_delayslot

# and so on for: jr, jal, bne, bltz, bgezal, bczt...

\_done: mfc0 $k1, $14 # reload EPC into $k1

addiu $k1, $k1, 4 # Do not reexecute fault

instr

jr $k1

rfe # done in delay-slot of jr

\_delayslot: mfc0 $k1, $14 # reload EPC into $k1

addiu $k0, $k1, -4 # $k0 = EPC - 4

addiu $k1, $k1, 4 # $k1 = EPC + 4

jr $k0 # poke at branching instr

rfe

.kdata

save0: .word 0

save1: .word 0

8. Rewrite the base exception handler that comes with SPIM (described in App A.7) to look for arithmetic overflow exception. For an arithmetic overflow exception inspect the arithmetic instruction that caused the exception through EPC; modify the contents of the dest register of this instruction as follows. If the sign of the current value in dest register Rd is 1 (remember 1 with an overflow means that the actual result was meant to be positive) then write Rd← 0x7fffffff, else Rd← 0x80000000. Try out your new exception handler with SPIM on a small program that forces an arithmetic overflow. [15 points]

.ktext 0x80000180

mov $k1, $at # Save $at register

sw $a0, save0 # Handler is not re-entrant and can’t use

sw $a1, save1 # stack to save $a0, $a1

# Don’t need to save $k0/$k1

mfc0 $k0, $13 # Move Cause into $k0

srl $a0, $k0, 2 # Extract ExcCode field andi $a0, $a0, Oxf

bgtz $a0, done # Branch if ExcCode is Int (0)

mov $a0, $k0 # Move Cause into $a0

mfco $a1, $14 # Move EPC into $a1

jal print\_excp # Print exception error message

done: mfc0 $k0, $14 # Bump EPC

addiu $k0, $k0, 4 # Do not re-execute faulting instruction

mtc0 $k0, $14 # EPC

mtc0 $0, $13 # Clear Cause register

mfc0 $k0, $12 # Fix Status register

andi $k0, Oxfffd # Clear EXL bit

ori $k0, Ox1 # Enable interrupts

mtc0 $k0, $12

lw $a0, save0 # Restore registers

lw $a1, save1

mov $at, $k1

eret # Return to EPC

.kdata

save0: .word 0

save1: .word 0