Matthew Park

34017103

Assignment 3 – CPSC 313 Summer 2012

1.

Bug #1 found in test at Memory 300 (sBHazard)

What should be expected at the end of the test:

%eax = 17

%ecx = 18

%edx = 19

%ebx = 16

What showed up before the bug fix:

%eax = 16

%ecx = 16

%edx = 16

%ebx = 16

Erroneous Action:

The erroneous execution is that for the pipelineHazardControl, when coming across a data hazard, it is not checking the data hazard on the second register.

Solution:

For the if statement’s condition of the data hazard, which is checking only for the data hazard of the first register, I add it to check for the data hazard on the second register as well.

Original implementation:

Line 41:

**if** (isDataHazardOnReg (d.srcA.getValueProduced()))

After Fix:

**if** (isDataHazardOnReg (d.srcA.getValueProduced()) ||

isDataHazardOnReg(d.srcB.getValueProduced()))

Bug #2 found in test at Memory 400 (aLoadUse)

What should be expected at the end of the test:

%eax = 10

%ecx = 10

%edx = 10

%ebx = 30

%edi = 0x1000 or 4096

What showed up before the bug fix:

%eax = 10

%ecx = 10

%edx = 10

%ebx = 0

%edi = 0x1000 or 4096

Erroneous Action:

The erroneous execution is that for the helper function isDataHazardOnReg(**int** reg), it is not checking for the data hazard on register output port for dstM.

Solution:

For that function, I extended the return statement to also check dstM for the output registers on Execute, Memory, and Write-back stages.

Original implementation:

Line 32:

**return** reg != *R\_NONE* && (E.dstE.get() == reg || M.dstE.get() == reg || W.dstE.get() == reg);

After Fix:

**return** reg != *R\_NONE* && (E.dstE.get() == reg || M.dstE.get() == reg || W.dstE.get() == reg || E.dstM.get() == reg || M.dstM.get() == reg|| W.dstM.get() == reg);

Bug #3 found in test at Memory 700 (notTKJmp)

What should be expected at the end of the test:

%eax = 0

%ecx = 1

%edx = 1

%ebx = 0

%esp = 0

What showed up before the bug fix:

%eax = 0

%ecx = 1

%edx = 1

%ebx = 1

%esp = 0

Erroneous Action:

The erroneous execution is that for the pipelineHazardControl, when coming across a control hazard, the conditional jump is not doing a hazard control when a jump instruction is in the Execute stage.

Solution:

For the if statement’s condition of the cump control hazard, I added it to check and apply hazard control for the Execute stage as well.

Original implementation:

Line 49:

**else** **if** ((D.iCd.get()==*I\_JXX* && D.iFn.get()!=*C\_NC*))

After Fix:

**else** **if** ((D.iCd.get()==*I\_JXX* && D.iFn.get()!=*C\_NC*) || (E.iCd.get()==*I\_JXX* && E.iFn.get()!=*C\_NC*))

2.

CPI for sum.s

Cycles per Instruction (CPI) = total cycles / instructionRetired Cycles

= cCnt / iCnt

= 117 / 45

= 2.6 CPI

The CPI for sum.s is approximately 2.6 cycles per instruction.

CPI for max.s

Cycles per Instruction (CPI) = total cycles / instructionRetired Cycles

= cCnt / iCnt

= 236 / 98

= 2.4081… CPI

The CPI for max.s is approximately 2.4082 cycles per instruction.

CPI for heapsort-student.s

Cycles per Instruction (CPI) = total cycles / instructionRetired Cycles

= cCnt / iCnt

= 7796 / 3001

= 2.5978… CPI

The CPI for heapsort-student.s is approximately 2.5978 cycles per instruction.

**Time Spent on this Assignment: 3 hours**