Ahmed Saaif ahmeds7@rpi.edu

Sunday, March 1, 2020 4:40 PM

Problem 1:

a)

Answer: lw and sw are the only instructions in the table that need to access memory so 35%

b)

Answer: 80% of all cycles need the sign-extended input. Add is the only one that is not I-type so it does not need the sign-extended circuit.

Problem 2:

a)

Answer: In a pipelined system everything, ideally, runs in a perfect parallel. Thus the clock cycle will be determined by the slowest instruction. Which in this case is ID so 350ps.
In a non-pipelined system everything runs serially such that the clock cycle time will be the sum of all instructions or in this case 1,250ps.

b)

Answer: LW must use all instructions so in a pipelined system it can only do one instruction at a time. Which means that it will need 5 cycles so $5 * 350ps - 1750ps$.
In a non-pipelined system you can order the instructions so everything can be done in one clock cycle so it is $1250ps$

c)

Answer: The only one to split is ID because it is the longest stage. This means that the new longest instruction is MEM so the new clock cycle is 300ps

d)

Answer: Data Memory is only used by the lw and sw instructions so 15% is the utilization

e)

Answer: Write registers is used by lw and the alu instructions so 65% is the utilization

Problem 3

a)

Because there are dependencies, without forwarding there is always a risk of hazards. This occurs from the first statement to the second. And from the second to the third, etc.
Two instructions are needed because they are each written to and read from

Answer:
or r1, r2, r3
NOP
NOP
or r2, r1, r4
NOP
NOP
or r1, r1, r2
NOP
NOP
or r1, r1, r2

b)

With full-forwarding the hazard of a value not being updated for the next operation is removed. Since this was the only hazard we had, we do not need NOP codes

Answer:
or r1, r2, r3
or r2, r1, r4
or r1, r1, r2
or r1, r1, r2

c)

A normal or instruction takes 5 cycles, with pipelining it will take 8 to do all 4 instructions.
There are 6 NOP instructions which means there is a 6 cycle delay.
With full forwarding we need no delays. So 8 cycles

Answer:
$8 + 6 = 14 \rightarrow 14 * 250ps = 3500ps$ with no forwarding
$8 * 350ps = 2800ps$ with full forwarding.
There is a 20% speedup from no forwarding.

d)

With there is the ID/EX EX/MEM forwarding we can remove one NOP between so.

Answer:
or r1, r2, r3
NOP
or r2, r1, r4
NOP
or r1, r1, r2
NOP
or r1, r1, r2

e)

Answer: $8 + 3 = 11 \rightarrow 11 * 300ps = 3300ps$ with the partial forwarding
There is a 5.7% speedup as compared to no forwarding.

Problem 4

a)

Answer:
add r5,r2,r1
NOP
NOP
lw r3,4(r5)
lw r2,0(r2)
NOP
or r3,r5,r3
NOP
NOP
sw r3,0(r5)

b)

Forwarding avoids hazards so
Answer:
add r5,r2,r1
lw r3,4(r5)
lw r2,0(r2)
or r3,r5,r3
sw r3,0(r5)

c)

There isn't any place we can use the register r7 to our advantage because the registers are updated in each instruction. We can only swap instructions.

Answer:
add r5,r2,r1
lw r2,0(r2)
NOP
lw r3,4(r5)
NOP
NOP
or r3,r5,r3
NOP
NOP
sw r3,0(r5)

`

d)

This code should execute as normal but may have issues in other blocks of code. Without hazard detection, when reading from memory there is still a risk of running a bad piece of data into the rest of the code. There needs to be another delay when using values that are loaded from memory to ensure that it isn't any undesired data.

Problem 5

a)

The branch instruction is BEQ. It takes up 25% of the instructions. The accuracy of the Always Taken is 45% so the inaccuracy is 55%.
Answer: $3 * .25 * .55 = 0.4125$ extra CPI

b)

Same process as part A, except the inaccuracy is $1 - 0.55 = .45$
Answer: $3 * .25 * .45 = 0.3375$ extra CPI

c)

Same process as part A except that the inaccuracy $1 - .85 = .15$
Answer: $3 * .25 * .15 = 0.1125$ extra CPI

d)

Without conversion: $1 + 0.1125 = 1.1125\ CPI$
With conversion: $1 + (3 * .25 * .15 * .5) = 1.056\ CPI$
Answer: $speedup = \frac{1.1125}{1.056} = 1.054$ times faster

e)

Without conversion: $1.1125\ CPI$
With conversion we take an extra cycle: $1 + (1 + 3 * .15) * .25 * .5 = 1.181$
Answer: $speedup = \frac{1.1125}{1.181} = 0.94$ times faster. So in this case the extra cycle makes it slower.