Max Shi  
CS 383 C  
Professor Barbalace  
November 3, 2019  
I pledge my honor that I have abided by the Stevens Honor System.

Homework 3

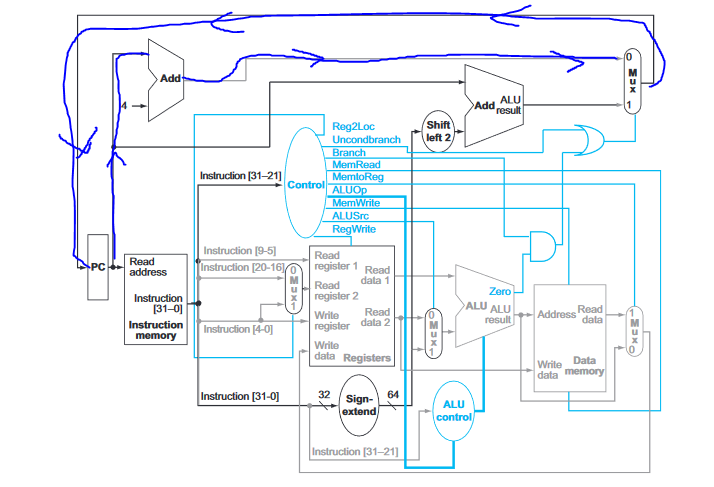
**4.5.1.**

Binary instruction: 1111 1000 0000 0001 0100 0000 0110 0010  
31:21 = 11111000000 = 1984 = STUR instruction  
20:12 = 000010100 = address (sign extend)

Sign extend whole instruction = extend 9 bits to 64 bits = (55 0’s) + 000010100  
Shift left two bits = (55 0’s) + 001010000

4.5.2. ALU control input = 0010

4.5.3. PC = PC + 4



4.5.5.

ALU Input – (55 0’s) + 000010100 and 00011  
Add Unit 1 – PC and 4  
Add Unit 2 – PC and (55 0’s) + 001010000

4.8.  
R/I type = 725ps, LDUR = 925ps, STUR = 880ps, CBZ = 735ps, B = 525 ps.  
Average time/instruction = 0.52\*725 + 0.25\*925 + 0.10\*880 + 0.11\*735 + 0.02\*525 = 787.6 ps  
Single cycle CPU = 925 ps.   
787.6/925 = 1.174, 17.4% speedup.

4.9.1.  
Clock cycle time with = 925+300 = = 1225ps.  
Clock cycle time without = 925 ps.

4.9.2.  
Assume 100 instructions.   
Without = 925\*100 = 92500 ps  
With = 1225\*95 = 116375 ps  
92500/116375 = 0.795 = 20.5% slowdown.

4.9.3.  
92500ps = 95\*(925+x)  
973.68=925+x  
x=48.68ps  
The ALU latency can be as high as 48.68ps to have no net performance impact.

4.16.1.  
Pipelined – as slow as the slowest portion = 350ps.  
Non-pipelined – sum of all parts = 250+350+150+300+200=1250

4.16.2.

LDUR instruction would take the same amount of time total – 1250ps.

4.16.3.

I would split ID, as it is the longest stage. Clock cycle rate = 300ps, as MEM is new longest stage.

4.16.4.

Data memory is STUR and LDUR = 20%+15% = 35%.

4.21.1.

Ignoring initial latency as the impact of this approaches 0 as number of instructions increases.

Let n = number of instructions.

Old pipeline –  
Total instructions – n + 0.4\*n = 1.4n  
250\*1.4n = 350n ps

New pipeline -   
Total instructions – n+0.05\*n = 1.05n  
300\*1.05n = 315n ps  
  
350/315 = 1.1111 = 11.1% speedup

4.21.2.  
350n = 300\*(1+x)n  
350 = 300+300x  
50 = 300x  
x = 50/300 = 0.166. 1/6 of the instructions can be NOPs in the forwarding pipeline before it becomes slower than the non-forwarded pipeline.

4.21.3.  
250 \* (1+x)n = 300\*(1 + y) n --- solve for y  
250 + 250x = 300 + 300y  
250x -50 = 300y  
y = (250x – 50) / 300.  
The maximum amount of NOPs that can remain in the forwarding pipeline is (250x – 50)/300, where x is the proportion of NOPs to instructions in the non-forwarded pipeline.

4.21.4.   
No because using the formula from the previous problem, setting x = 0.075 gets y=-0.104, which is impossible as there cannot be a negative proportion of NOPs.

4.22.1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CC | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| STUR | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |  |
| LDUR |  | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |
| SUB |  |  | IF | ID | EX | ME | WB |  |  |  |  |  |  |
| CBZ |  |  |  | STALL | STALL | STALL | IF | ID | EX | ME | WB |  |  |
| ADD |  |  |  |  |  |  |  | IF | ID | EX | ME | WB |  |
| SUB |  |  |  |  |  |  |  |  | IF | ID | EX | ME | WB |

4.22.2.

In general, reordering code does not help, as either the output of the code will change, which is undesirable, or at some point, the memory access will happen in the same step as the instruction fetch, which will create the same structural hazard, and the NOPs need to happen three times to break up the memory accesses and the instruction fetches.

4.25.1.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CC | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| LDUR | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |  |  |  |  |
| LDUR |  | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |  |  |  |
| ADD |  |  | IF | STALL | ID | EX | ME | WB |  |  |  |  |  |  |  |  |
| SUBI |  |  |  |  | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |
| CBNZ |  |  |  |  |  | IF | ID | EX | ME | WB |  |  |  |  |  |  |
| LDUR |  |  |  |  |  |  | IF | ID | EX | ME | WB |  |  |  |  |  |
| LDUR |  |  |  |  |  |  |  | IF | ID | EX | ME | WB |  |  |  |  |
| ADD |  |  |  |  |  |  |  |  | IF | STALL | ID | EX | ME | WB |  |  |
| SUBI |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | ME | WB |  |
| CBNZ |  |  |  |  |  |  |  |  |  |  |  | IF | ID | EX | ME | WB |

4.27.1.

ADD X5, X2, X1  
LDUR X3, [X5, #4]  
LDUR X2, [X2, #0]  
ORR X3, X5, X3  
STUR X3, [X5, #0]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CC | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| ADD | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |  |  |  |  |
| LDUR |  | NOP | NOP | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |  |
| LDUR |  |  |  |  | IF | ID | EX | ME | WB |  |  |  |  |  |  |  |
| ORR |  |  |  |  |  | NOP | IF | ID | EX | ME | WB |  |  |  |  |  |
| STUR |  |  |  |  |  |  |  | NOP | NOP | IF | ID | EX | ME | WB |  |  |

4.27.2.

There is no possible way to reorder the code to reduce the amount of NOPs.

4.27.3.

It outputs correctly. The only NOPs we used in this are to prevent data hazards, not structural hazards, so the code executes correctly with only forwarding implemented.

4.29.1.

Accuracy of taken = 3/5 = 60%

Accuracy of not taken = 2/5 = 40%.

4.29.2.

T = miss

NT = hit

T = miss

T = miss

1/4 = 25%