Kenneth Neads

4/21/2019

Computer Architecture Assignment 6

2. Add $3, $4, $2
3. Sub $5, $3, $1
4. Lw $6, 200($3)
5. Add $7, $3, $6

B is dependent on A -> fixed by fowarding

C is dependent on A -> fixed by forwarding

D is dependent on A and C -> fixed by stalling and forwarding

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Clock Cycle | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  |
| A. | IF | ID | EXE | MEM | RB |  |  |  |  |
| B. |  | IF | ID | EXE | MEM | RB |  |  |  |
| C. |  |  | IF | ID | EXE | MEM | RB |  |  |
|  |  |  |  | Stall |  |  |  |  |  |
| D. |  |  |  |  | IF | ID | EXE | MEM | RB |

2. If we can’t forward, it’ll be like C. and D. above, except with an extra stall, therefor it will have 2 stalls before and after the commands. With a million instructions and pipelining, the average load command will take 1 cycle, the average add command will take 1 cycle, and in total there are 4 stalls, so 6 cycles total. 6 cycles / 2 instructions = 3 cycles per instruction
3. With forwarding, there will still be a bubble between load and add like C and D above, but I don’t know if anything will change with the other spaces. I think it’ll be 4 cycles? 2 stalls, and 2 instruction/cycle so cpi will be 2?
5. Ideal pipeline CPI is 1, actual cpi will be 0.15 + 0.20 + 1 = 1.35. Increase is 0.35 cpi
6. (1GHz/150MHz) = 6.6666 6.6666 \* (1/1.35) = 4.94 speedup
8. I have no idea how we can remove this bubble? Maybe if we somehow skip the bubble altogether or make a route for the data to skip a cycle/step? Otherwise there would be no way to stop the bubble? OR we force a signal to act as a clock cycle to skip ahead a step????
10. Always taken Correct = 15/25 = 60%
    1. T-T-T 3/3
    2. T-T-T-T 0/4
    3. T-T-T-T-T-T 3/6
    4. T-T-T-T-T 4/5
    5. T-T-T-T-T-T-T 5/7
11. Always not taken Correct = 10/25 = 40%
    1. N-N-N 0/3
    2. N-N-N-N 4/4
    3. N-N-N-N-N-N 3/6
    4. N-N-N-N-N 1/5
    5. N-N-N-N-N-N-N 2/7
12. 1-bit predictor, initialized to predict taken Correct = 13/25 = 52%
    1. T-T-T 3/3
    2. T-N-N-N 3/4
    3. T-T-N-T-N-T 1/6
    4. T-T-T-T-N 3/5
    5. T-T-T-N-T-T-T-N 3/7
13. 2-bit predictor, initialized to weakly predict taken (1,0) Correct = 18/25 = 72%
    1. T-T-T 3/3
    2. T-N-N-N 3/4
    3. T-T-T-T-T-T 3/6
    4. T-T-T-T-T 4/5
    5. T-T-T-T-T-T-T 5/7