**CDA-5106 Assignment 2**

3.15

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **Instruction** | **Issues** | **Executes** | **Memory Access** | **CDB Access** | **Comment** |
| 1 | Fld f2,0(x1) | 1 | 2 | 2 | 3 | Load the base address of X2 into F2 |
| 1 | Fmul f4,f2,f0 | 2 | 4 | 4 | 19 | The instruction has to wait till the previous load instruction completes since the instruction is using F2 |
| 1 | Fld f6,0(x2) | 3 | 4 | 4 | 5 | Load the base address of X2 into F6 |
| 1 | Fadd f6,f4,f6 | 4 | 20 | 20 | 30 | The instruction only starts execution when fmul completes because of f6 dependency |
| 1 | Fsd f6,0(x2) | 5 | 31 | 31 | NA | Wait for F6 to be written to before storing in buffer |
| 1 | Addi x1,x1,#8 | 6 | 7 | 7 | 8 | No wait |
| 1 | Addi x2,x2,#8 | 7 | 8 | 8 | 9 | No wait |
| 1 | Sltu x3,x1,x4 | 8 | 9 | 9 | 10 | No wait |
| 1 | Bnez x3, foo | 9 | 11 | 11 | NA | No wait |
| 2 | Fld f2,0(x1) | 10 | 12 | 12 | 13 | Wait for bnez to finish |
| 2 | Fmul f4,f2,f0 | 11 | ` |  |  |  |
| 2 | Fld f6,0(x2) | 12 |  |  |  |  |
| 2 | Fadd f6,f4,f6 | 13 |  |  |  |  |
| 2 | Fsd f6,0(x2) | 14 |  |  |  |  |
| 2 | Addi x1,x1,#8 | 15 |  |  |  |  |
| 2 | Addi x2,x2,#8 | 16 |  |  |  |  |
| 2 | Sltu x3,x1,x4 | 17 |  |  |  |  |
| 2 | Bnez x3, foo | 18 |  |  |  |  |
| 3 | Fld f2,0(x1) | 19 |  |  |  |  |
| 3 | Fmul f4,f2,f0 | 20 |  |  |  |  |
| 3 | Fld f6,0(x2) | 21 |  |  |  |  |
| 3 | Fadd f6,f4,f6 | 22 |  |  |  |  |
| 3 | Fsd f6,0(x2) | 23 |  |  |  |  |
| 3 | Addi x1,x1,#8 | 24 |  |  |  |  |
| 3 | Addi x2,x2,#8 | 25 |  |  |  |  |
| 3 | Sltu x3,x1,x4 | 26 |  |  |  |  |
| 3 | Bnez x3, foo | 27 |  |  |  |  |

3.16

1. LD F1 X
2. LD F2 Y
3. ADD F4 F1 Z
4. MUL F3 F1 F2
5. LD F2 Z
6. MUL F3 F3 F2
7. ADD F4 F5 F6
8. ADD F4 F1 F2

Since the MUL instructions take 15 cycles, and the second instruction (Line 6) is dependent on the first one (Line 4), it will begin execution at the 22nd clock cycle. In the meantime, the ADD instructions can begin execution since there is no data dependence between them and the MUL instructions. However, each of the ADD instructions is dependent on each other so they cannot execute concurrently.

We assume that Load operations take 2 cycles. We also assume that X, Y and Z are already present in the registers.

There will be CDB contention in the 38th cycle when instructions 6 and 8 complete their execution at the same time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction number | Issue cycle | Starting cycle | Finishing cycle | Write back cycle |
| 1 | 1 | 2 | 4 | 5 |
| 2 | 2 | 3 | 5 | 6 |
| 3 | 3 | 5 | 15 | 16 |
| 4 | 4 | 6 | 21 | 22 |
| 5 | 5 | 7 | 9 | 10 |
| 6 | 6 | 22 | 37 | 38 |
| 7 | 7 | 16 | 26 | 27 |
| 8 | 8 | 27 | 37 | 38 |

3.17

3.18

Stalls due to Branch Target Buffer (BTB) = (Stall due to buffer miss) + (Stall due to branch take in buffer) + (Stall due to branch misprediction in buffer)

Probability of buffer miss (PMiss) = Branch Frequency \* Miss rate

= 15% \* 10% = 1.5%

Probability of branch taken (PTaken) = Branch Frequency \* Hit rate \* Accuracy

= 15% \* 90% \* 90% = 12.1%

Probability of branch misprediction (PMisprediction) = Branch Frequency \* Hit rate \* (100-Accuracy)

= 15% \* 90% \* 10\* = 1.2%

Total stall = PMiss\*penalty + PTaken\*0 + PMisprediction\*penalty

= 1.5%\*3 + 0 + 1.2\*4

= 9.3%

= 1 + (9.3/100) = **1.093 CPI**

For a processor with a fixed two cycle branch penalty,

Stall = Branch frequency \* penalty

= 15% \* 2

= 0.3

= 1.3

**Speed up = 1.3/1.093 ~ 1.2**