Exercise Session 6

Complex Pipeline, Dynamic Branch Prediction, Scoreboard/Tomasulo

Advanced Computer Architectures

Politecnico di Milano

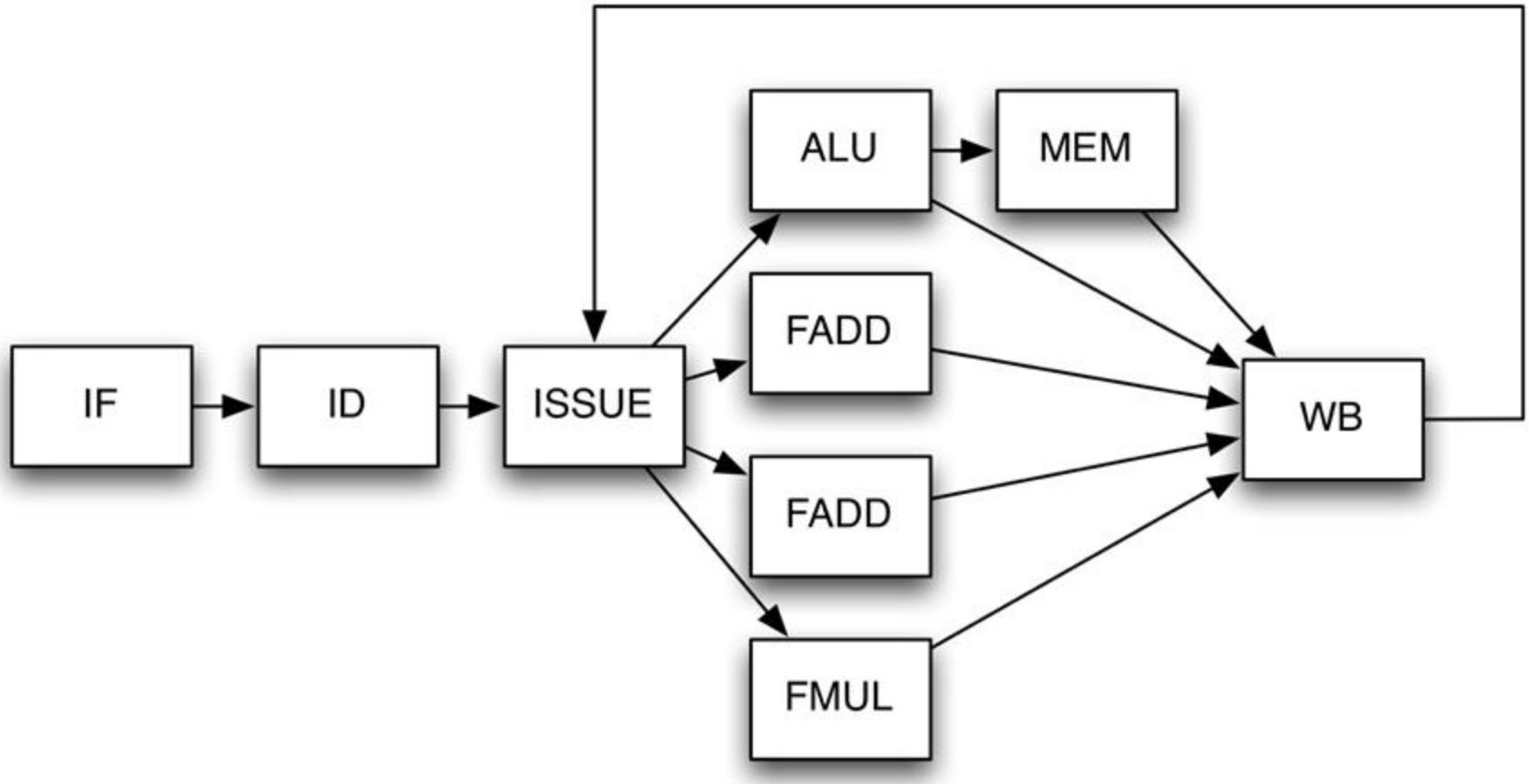
April 23rd, 2025

Alessandro Verosimile <alessandro.verosimile@polimi.it>

POLITECNICO MILANO 1863

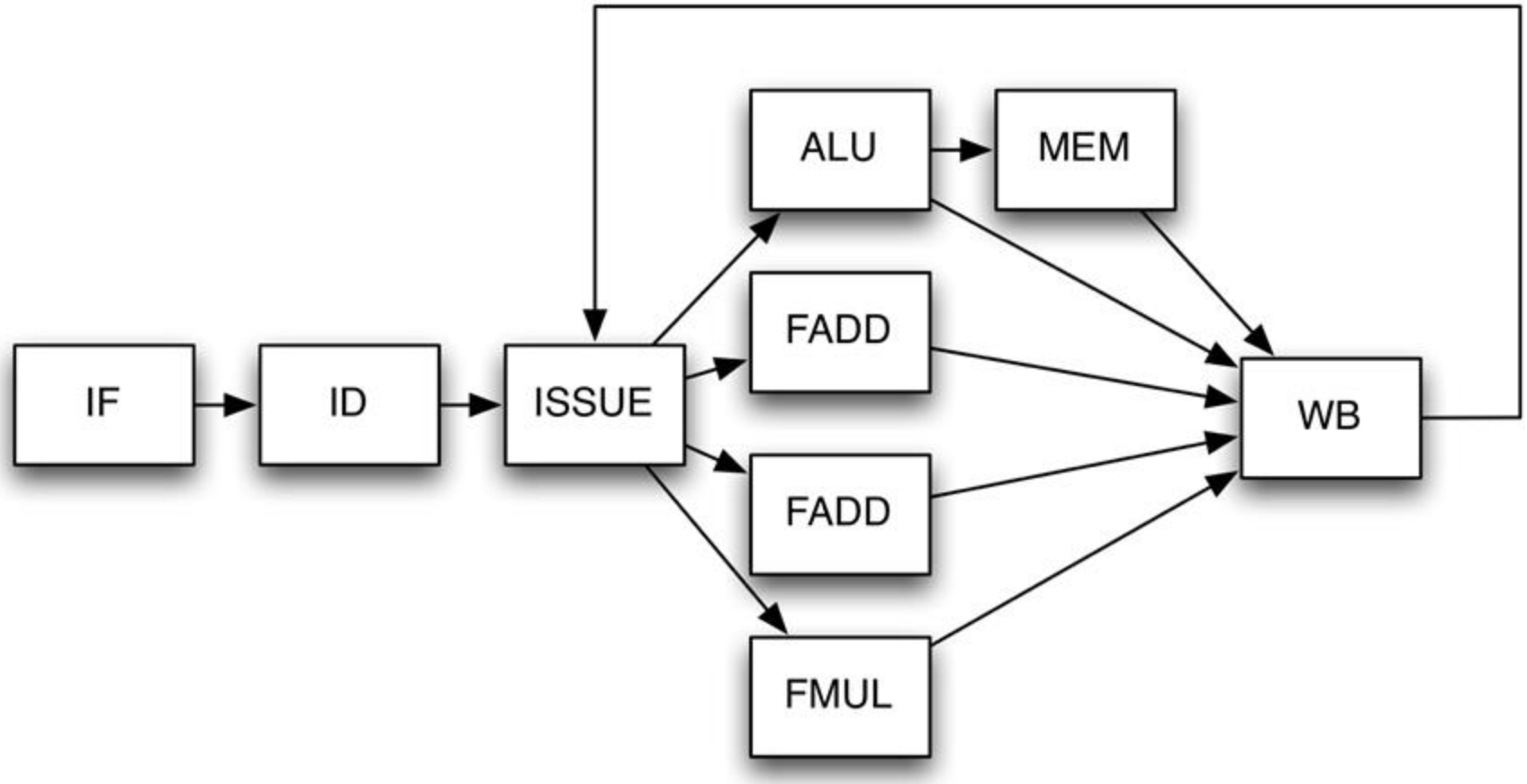
POLITECNICO MILANO 1863 NECST laboratory

Exe: Complex Pipeline

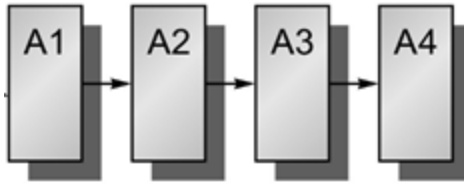


Exe: Complex Pipeline

In this problem we will examine the execution of a code segment on the following single-issue out-of-order processor:

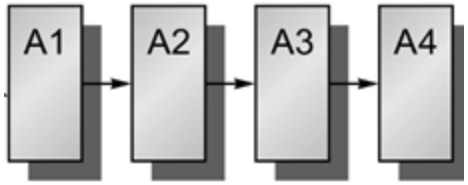


You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 2 cycles (includes time in ALU)
- Floating-point add instructions take 2 cycles
- Floating-point multiply instructions take 3 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage

You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 2 cycles (includes time in ALU)
- Floating-point add instructions take 2 cycles
- Floating-point multiply instructions take 3 cycles
- There is no register renaming. No forwarding
- Instructions are fetched, decoded and issued in order
- The ISSUE stage is a buffer of unlimited length that holds instructions waiting to start execution
- An instruction will only enter the issue stage if it does not cause a WAR or WAW hazard
- Only one instruction can be issued at a time, and in the case multiple instructions are ready, the oldest one will go first
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage

Exe Complex Pipeline: the Code

```
LOOP:I1: LD F1, 0 (R2)
     I2: MULTD F2, F1, F1
     I3: ADDD F3, F1, F5
     I4: MULTD F2, F3, F1
     I5: SUBD F5, F1, F5
     I6: SUBI R2, R2, 4
     I7: BNEZ R2, LOOP
```

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

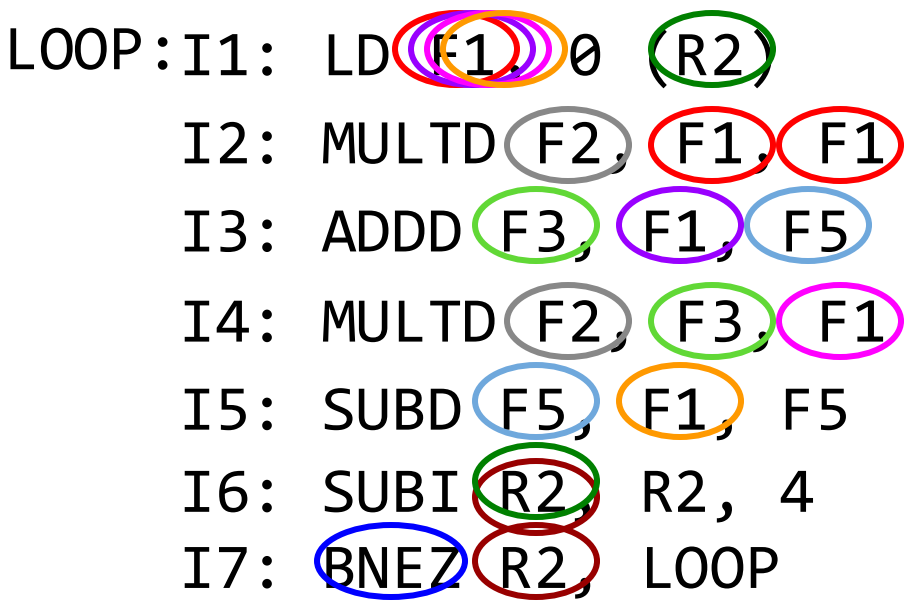
Exe Complex Pipeline: the Conflicts

```
LOOP:I1:  LD   F1, 0  (R2)
     I2:  MULTD F2, F1, F1
     I3:  ADDD  F3, F1, F5
     I4:  MULTD F2, F3, F1
     I5:  SUBD  F5, F1, F5
     I6:  SUBI  R2, R2, 4
     I7:  BNEZ  R2, LOOP
```

RAW F1 I1-I2
RAW F1 I1-I3
RAW F1 I1-I4
RAW F1 I1-I5
RAW F3 I3-I4
RAW R2 I6-I7
WAW F2 I2-I4
WAR F5 I3-I5
WAR R2 I1-I6
CNTRL

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

Exe Complex Pipeline: the Arch.



RAW F1 I1-I2
RAW F1 I1-I3
RAW F1 I1-I4
RAW F1 I1-I5
RAW F3 I3-I4
RAW R2 I6-I7
WAW F2 I2-I4
WAR F5 I3-I5
WAR R2 I1-I6
CNTRL

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 0

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)																				
2	MULTD F2,F1,F1																				RAW F1 I1-I2
3	ADDD F3,F1,F5																				RAW F1 I1-I3
4	MULTD F2,F3,F1																				RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5																				RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4																				WAR R2 I1-I6
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 1

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F																			
2	MULTD F2,F1,F1																				RAW F1 I1-I2
3	ADDD F3,F1,F5																				RAW F1 I1-I3
4	MULTD F2,F3,F1																				RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5																				RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4																				WAR R2 I1-I6
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 2

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D																		
2	MULTD F2,F1,F1		F																		RAW F1 I1-I2
3	ADDD F3,F1,F5																				RAW F1 I1-I3
4	MULTD F2,F3,F1																				RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5																				RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4																				WAR R2 I1-I6
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 3

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS																	
2	MULTD F2,F1,F1		F	D																	RAW F1 I1-I2
3	ADDD F3,F1,F5			F																	RAW F1 I1-I3
4	MULTD F2,F3,F1																				RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5																				RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4																				WAR R2 I1-I6
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 4

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1																
2	MULTD F2,F1,F1		F	D	IS s																RAW F1 I1-I2
3	ADDD F3,F1,F5			F	D																RAW F1 I1-I3
4	MULTD F2,F3,F1				F																RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5																				RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4																				WAR R2 I1-I6
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 5

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2															
2	MULTD F2,F1,F1		F	D	IS s	IS s															RAW F1 I1-I2
3	ADDD F3,F1,F5			F	D	IS s															RAW F1 I1-I3
4	MULTD F2,F3,F1				F	D s															RAW F1 I1-I4 RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5					F s															RAW F1 I1-I5 WAR F5 I3-I5
6	SUBI R2,R2,4																				WAR R2 I1-I6
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 6

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS														~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s														~~RAW F1 I1-I3~~
4	MULTD F2,F3,F1				F	D s	D s														~~RAW F1 I1-I4~~ RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5					F s	F s														~~RAW F1 I1-I5~~ WAR F5 I3-I5
6	SUBI R2,R2,4																				~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 7

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1													~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS													~~RAW F1 I1-I3~~
4	MULTD F2,F3,F1				F	D s	D s	D s													~~RAW F1 I1-I4~~ RAW F3 I3-I4 WAW F2 I2-I4
5	SUBD F5,F1,F5					F s	F s	F s													~~RAW F1 I1-I5~~ WAR F5 I3-I5
6	SUBI R2,R2,4																				~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 9

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3											~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2											~~RAW F1 I1-I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D											~~RAW F1 I1-I4~~ RAW F3 I3-I4 ~~WAW F2 I2-I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F											~~RAW F1 I1-I5~~ WAR F5 I3-I5
6	SUBI R2,R2,4																				~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 10

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s										~~RAW F1 I1-I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s										~~RAW F1 I1-I4~~ RAW F3 I3-I4 ~~WAW F2 I2-I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D										~~RAW F1 I1-I5~~ WAR F5 I3-I5
6	SUBI R2,R2,4										F										~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP																				RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 11

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1 I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS									~~RAW F1 I1-I4~~ ~~RAW F3 I3 I4~~ ~~WAW F2 I2 I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s									~~RAW F1 I1 I5~~ ~~WAR F5 I3 I5~~
6	SUBI R2,R2,4										F	D									~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP											F									RAW R2 I6-I7
8	(New Instruction)																				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 12

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1 I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS	E1								~~RAW F1 I1-I4~~ ~~RAW F3 I3 I4~~ ~~WAW F2 I2 I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s	IS								~~RAW F1 I1 I5~~ ~~WAR F5 I3 I5~~
6	SUBI R2,R2,4										F	D	IS s								~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP											F	D								RAW R2 I6-I7
8	(New Instruction)												F s								CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 14

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1 I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS	E1	E2	E3						~~RAW F1 I1-I4~~ ~~RAW F3 I3 I4~~ ~~WAW F2 I2 I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s	IS	E1	E2						~~RAW F1 I1 I5~~ ~~WAR F5 I3 I5~~ Structural on WB
6	SUBI R2,R2,4										F	D	IS s	IS	E1						~~WAR R2 I1-I6~~
7	BNEZ R2, LOOP											F	D	IS s	IS s						RAW R2 I6-I7
8	(New Instruction)												F s	F s	F s						CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 15

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1-I3~~ Structural on ~~WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS	E1	E2	E3	W					~~RAW F1 I1-I4~~ ~~RAW F3 I3-I4~~ ~~WAW F2 I2-I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s	IS	E1	E2	E2 s					~~RAW F1 I1-I5~~ ~~WAR F5 I3-I5~~ Structural on WB
6	SUBI R2,R2,4										F	D	IS s	IS	E1	E1 s					~~WAR R2 I1-I6~~ Structural on WB
7	BNEZ R2, LOOP											F	D	IS s	IS s	IS s					RAW R2 I6-I7
8	(New Instruction)												F s	F s	F s	F s					CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 16

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1-I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS	E1	E2	E3	W					~~RAW F1 I1-I4~~ ~~RAW F3 I3-I4~~ ~~WAW F2 I2-I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s	IS	E1	E2	E2 s	W				~~RAW F1 I1-I5~~ ~~WAR F5 I3-I5~~ ~~Structural on WB~~
6	SUBI R2,R2,4										F	D	IS s	IS	E1	E1 s	E1 s				~~WAR R2 I1-I6~~ Structural on WB
7	BNEZ R2, LOOP											F	D	IS s	IS s	IS s	IS s				RAW R2 I6-I7
8	(New Instruction)												F s	F s	F s	F s	F s				CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 17

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1 I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS	E1	E2	E3	W					~~RAW F1 I1-I4~~ ~~RAW F3 I3 I4~~ ~~WAW F2 I2 I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s	IS	E1	E2	E2 s	W				~~RAW F1 I1 I5~~ ~~WAR F5 I3 I5~~ ~~Structural on WB~~
6	SUBI R2,R2,4										F	D	IS s	IS	E1	E1 s	E1 s	W			~~WAR R2 I1-I6~~ ~~Structural on WB~~
7	BNEZ R2, LOOP											F	D	IS s	IS s	IS s	IS s	IS			~~RAW R2 I6 I7~~
8	(New Instruction)												F s	F s	F s	F s	F s	F s			CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 2 cycles
FP ADD: 2 cycles
FP MULT: 3 cycles

CC 19

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	Notes
1	LOOP: LD F1,0(R2)	F	D	IS	E1	E2	W														
2	MULTD F2,F1,F1		F	D	IS s	IS s	IS	E1	E2	E3	W										~~RAW F1 I1-I2~~
3	ADDD F3,F1,F5			F	D	IS s	IS s	IS	E1	E2	E2 s	W									~~RAW F1 I1-I3~~ ~~Structural on WB~~
4	MULTD F2,F3,F1				F	D s	D s	D s	D s	D	IS s	IS	E1	E2	E3	W					~~RAW F1 I1-I4~~ ~~RAW F3 I3-I4~~ ~~WAW F2 I2-I4~~
5	SUBD F5,F1,F5					F s	F s	F s	F s	F	D	IS s	IS	E1	E2	E2 s	W				~~RAW F1 I1-I5~~ ~~WAR F5 I3-I5~~ ~~Structural on WB~~
6	SUBI R2,R2,4										F	D	IS s	IS	E1	E1 s	E1 s	W			~~WAR R2 I1-I6~~ ~~Structural on WB~~
7	BNEZ R2, LOOP											F	D	IS s	IS s	IS s	IS s	IS	E1	W	~~RAW R2 I6-I7~~
8	(New Instruction)												F s	F s	F s	F s	F s	F s	F	D	~~CNTRL~~

Recall: Pipeline performance

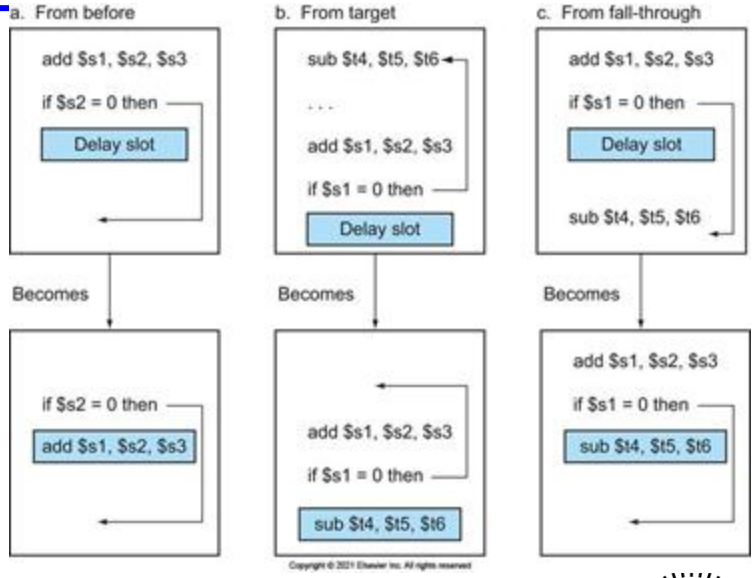
Pipeline CPI = Ideal pipeline CPI + Structural Stalls + Data Hazard Stalls + Control Stalls

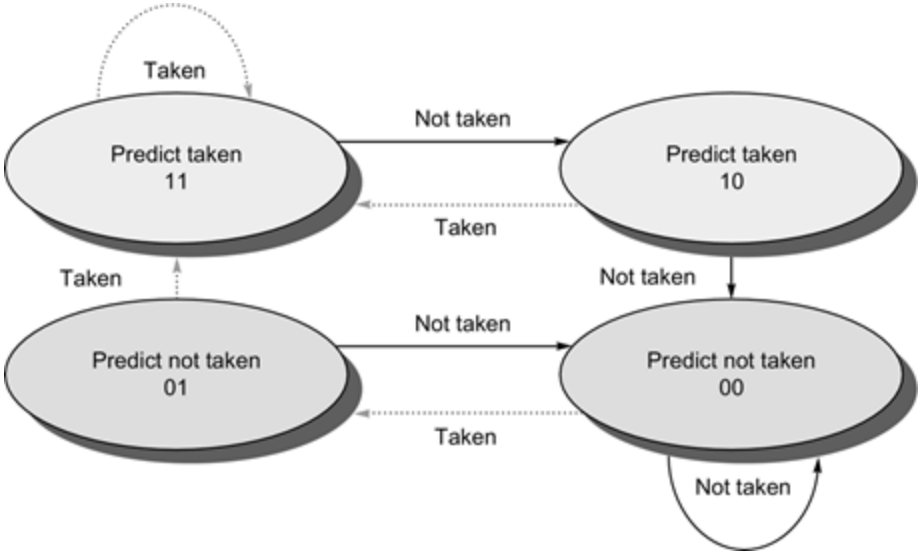
- **Ideal pipeline CPI**: measure of the maximum performance attainable by the implementation
- **Structural hazards**: HW cannot support this combination of instructions
- **Data hazards**: Instruction depends on result of prior instruction still in the pipeline
- **Control hazards**: Caused by delay between the fetching of instructions and decisions about changes in control flow (branches, jumps, exceptions)

Prediction: a.k.a. deal with Control Hazard

Branch vanguard: decomposing branch functionality into prediction and resolution instructions

The IBM z15 High Frequency Mainframe Branch Predictor





Dynamic Branch Predictor

- Describe (the answer has to be effectively supported) a 1-BHT and a 2-BHT able to execute the following assembly code (R0 is set to 1, R1 is set to 300)

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

- The obtained result, in terms of mispredictions, is inline with theoretical characteristics of the two predictors? Please effectively support your answer.

A First Consideration

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

How many iterations?

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300

How many iterations?

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300

LOOP2
@T0 → 3300 / 3 = 1100

How many iterations?

```
LOOP:   LD   F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300

LOOP2
@T0 → 3300 / 3 = 1100

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

How many iterations?

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300

LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

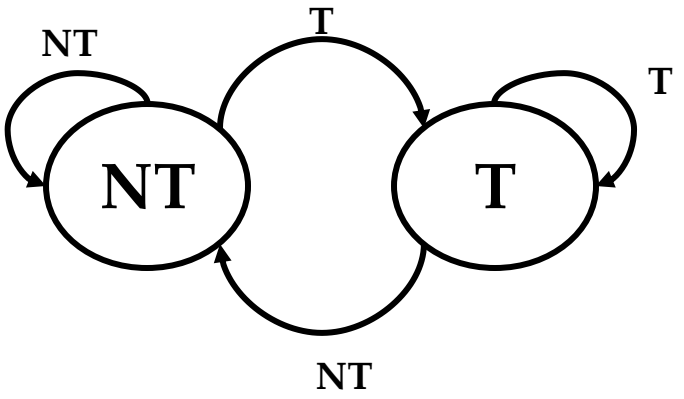
LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

1bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

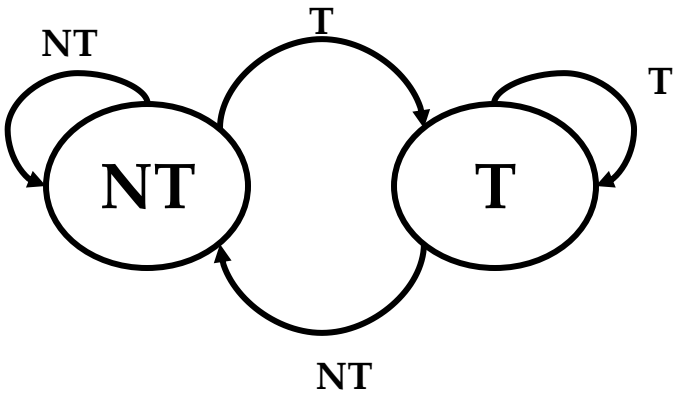
R0 is set to 1
R1 is set to 300

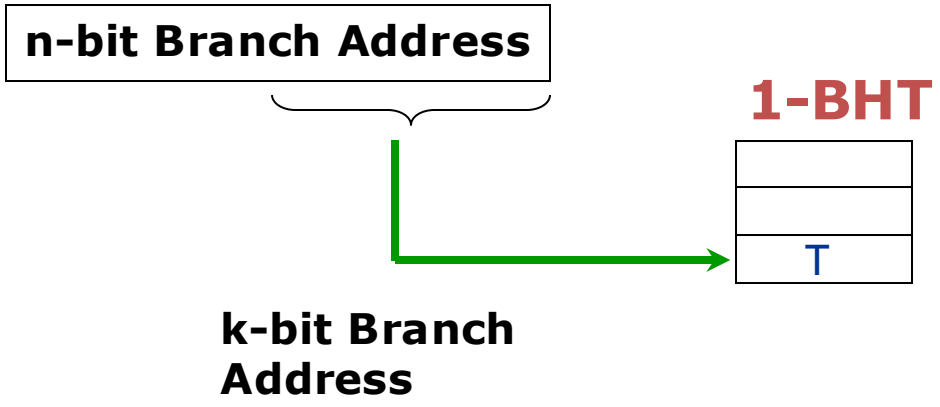


1bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



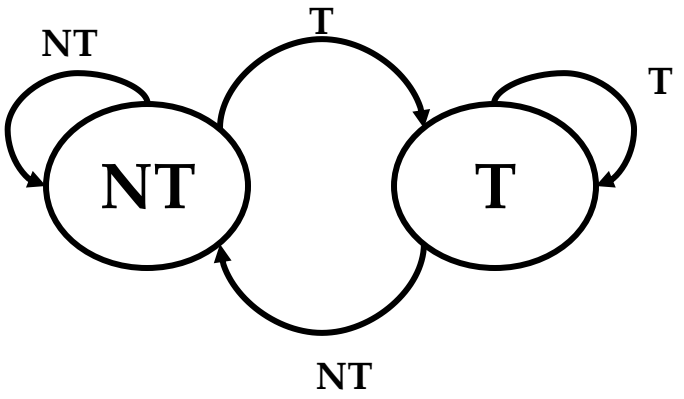


k-bit Branch Address:
Collide
Not collide

1bit - BHT - Not Collide

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



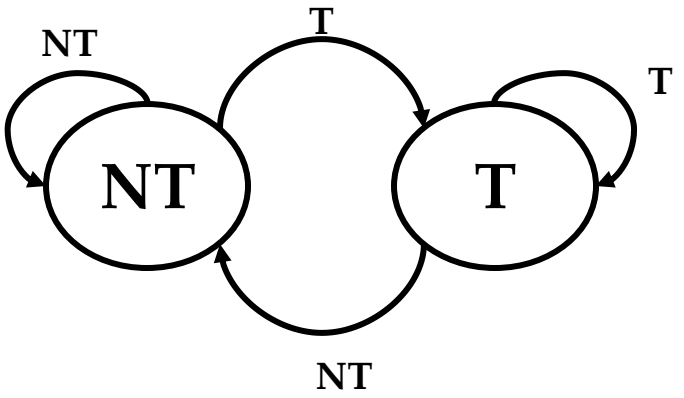
Let us consider that the branch addresses do not collide

	1-BHT	1-BHT	1-BHT	1-BHT
LOOP:	T	T	NT	NT
LOOP2:	T	NT	T	NT

1bit - BHT Misprediction

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



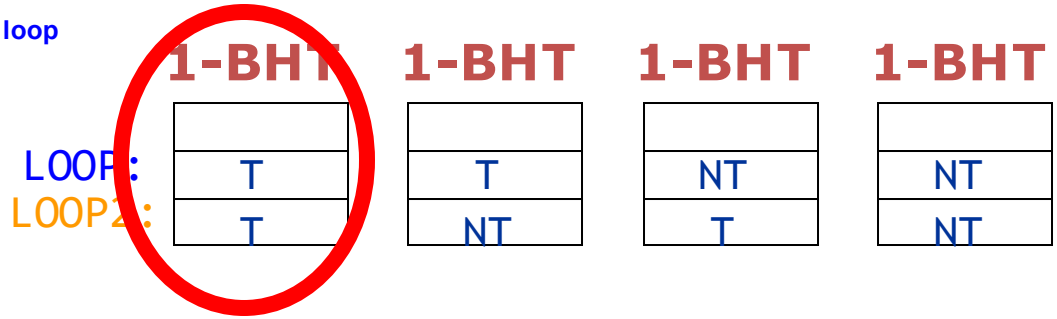
LOOP2
@T0 → 3300 / 3 = 1100

LOOP

@Ti 3000 / 3 = 1000 iterations

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do not collide

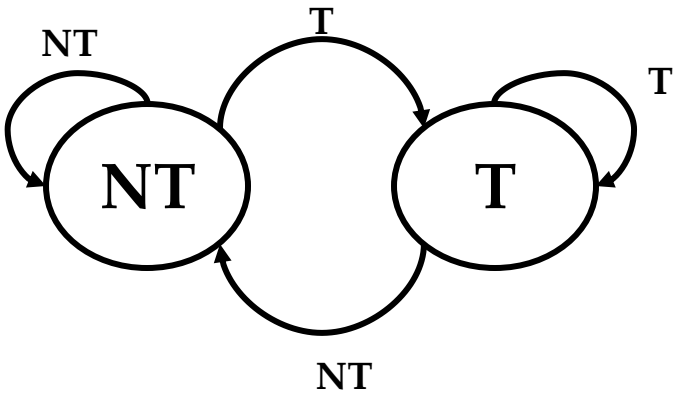


LOOP → NO misprediction
LOOP2 → single misprediction at the end of the loop @T0 + 2 misprediction x iteration begin and end of the loop

1bit - BHT Misprediction

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



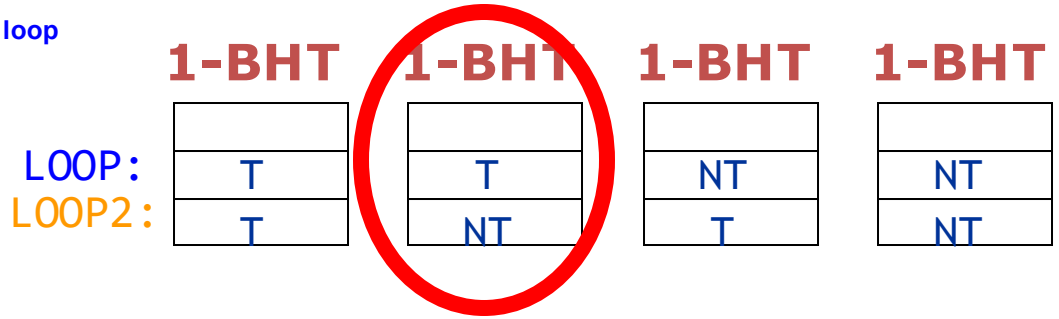
LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do not collide

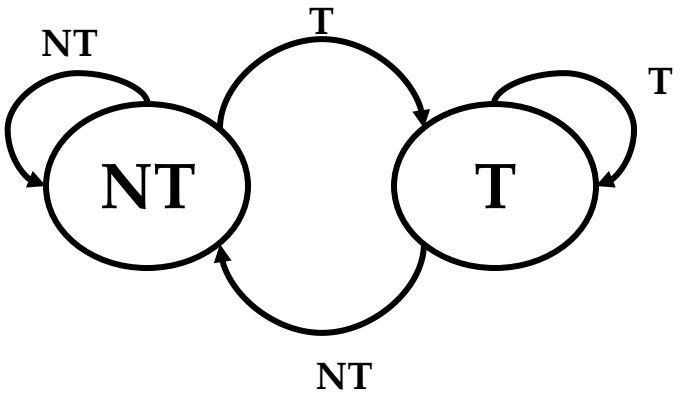


LOOP → NO misprediction
LOOP2 → two misprection: beginning and end of the loop

1bit - BHT Misprediction

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



LOOP2
@T0 → 3300 / 3 = 1100

LOOP

@Ti 3000 / 3 = 1000 iterations

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do not collide

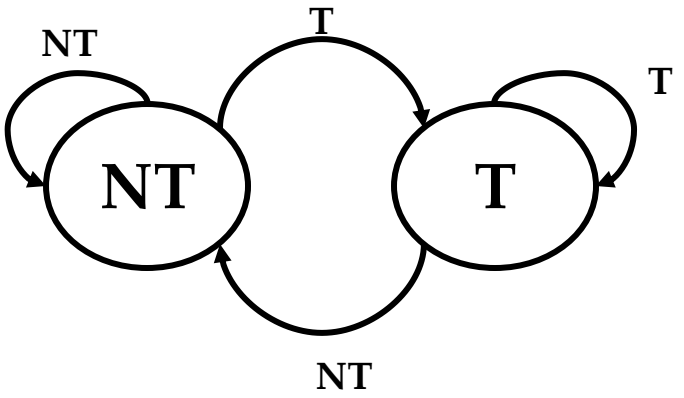
	1-BHT	1-BHT	1-BHT	1-BHT
LOOP:	T	T	NT	NT
LOOP2:	T	NT	T	NT

LOOP → only initial misprediction
LOOP2 → single misprediction at the end of the loop @T0 + 2 misprediction x iteration begin and end of the loop

1bit - BHT Misprediction

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do not collide

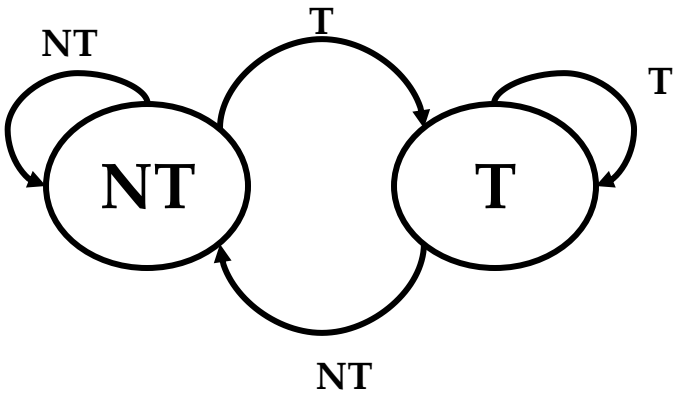
	1-BHT	1-BHT	1-BHT	1-BHT
LOOP:	T	T	NT	NT
LOOP2:	T	NT	T	NT

LOOP → only initial misprediction
LOOP2 → two misprection: beginning and end of the loop

1bit - BHT - Collision

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



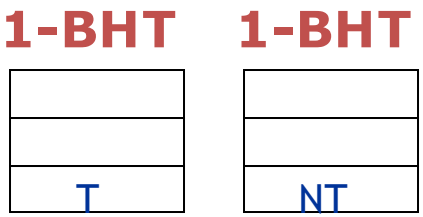
LOOP2
@T0 → 3300 / 3 = 1100

LOOP

@Ti 3000 / 3 = 1000 iterations

1 - 2 = -1 =/= 0!!!! → ∞ loop

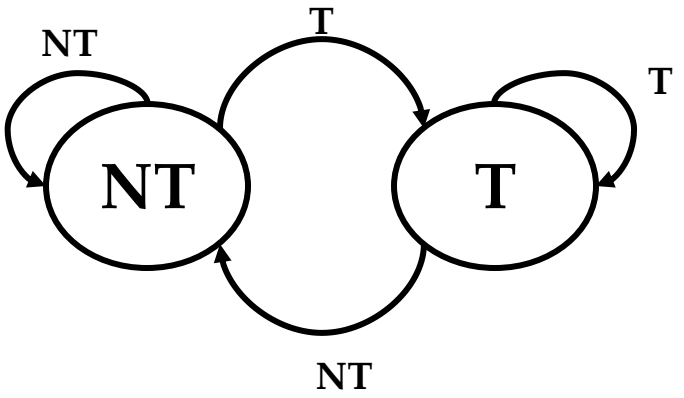
Let us consider that the branch addresses do collide



1bit - BHT - Misprediction

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



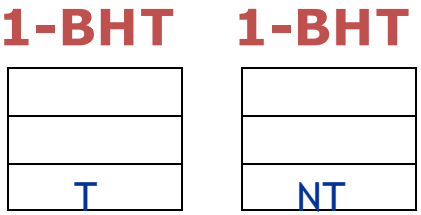
LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do collide



LOOP2 → single misprediction at the end of the loop
LOOP → 100% failure rate

LOOP2 → two initial misprection, then end of the loop
LOOP → 100% failure rate

2bit - BHT

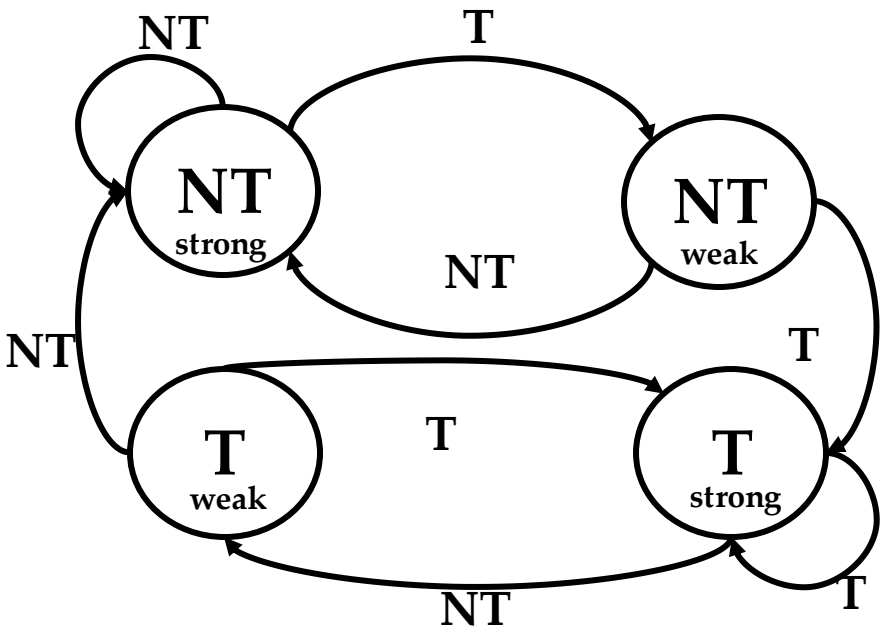
```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300

2bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

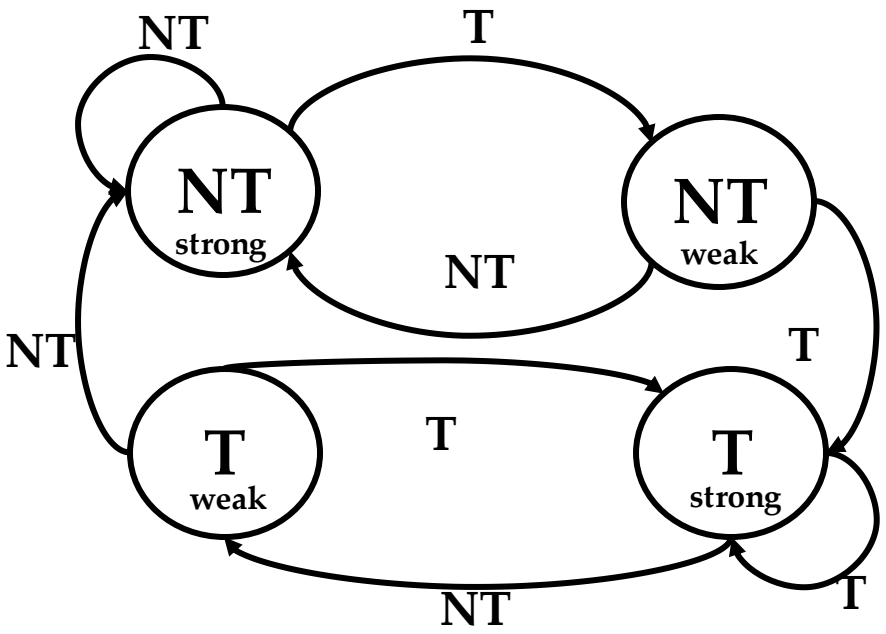
LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

2bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



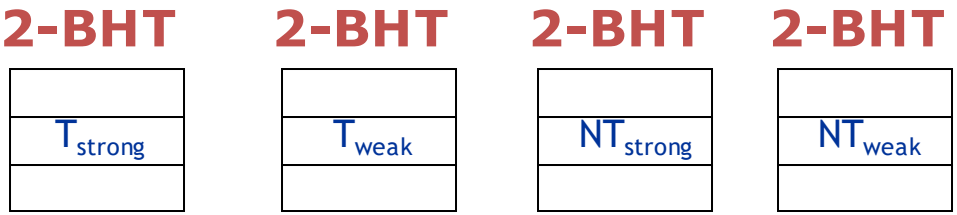
LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

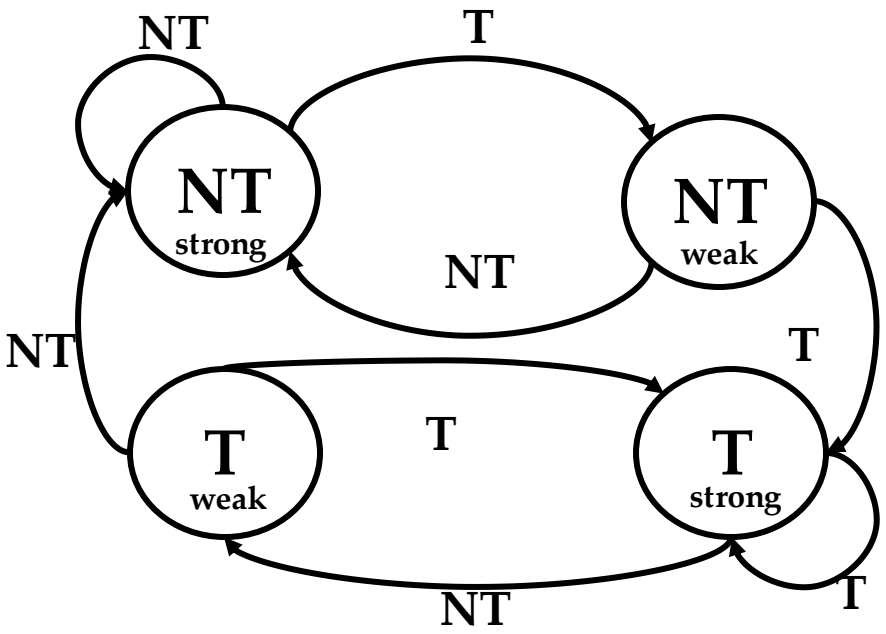
Let us consider that the branch addresses do collide



2bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



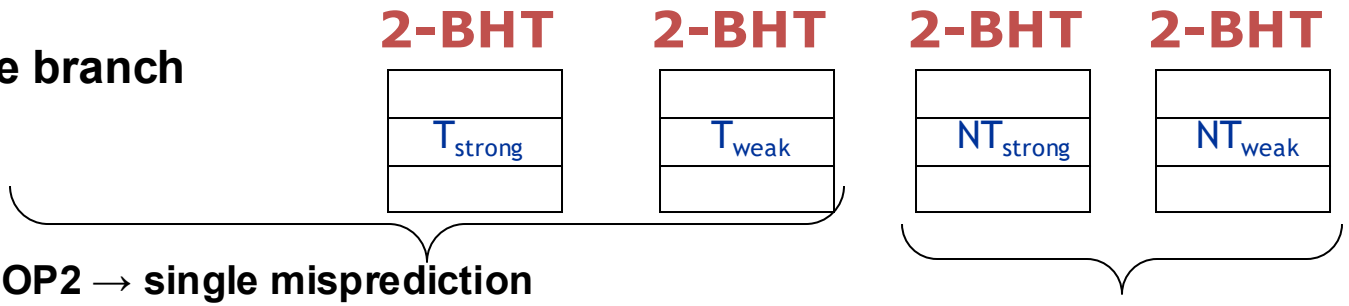
LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do collide



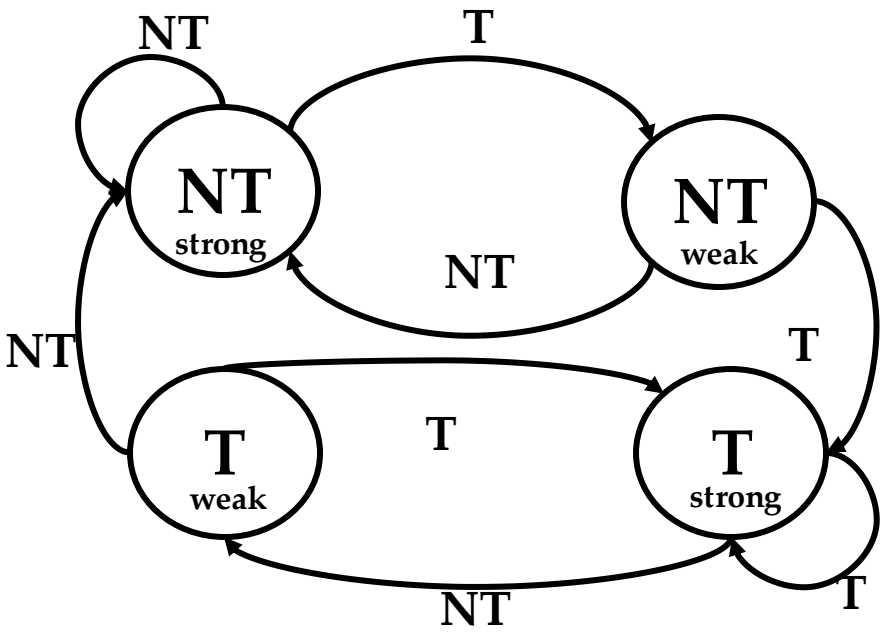
LOOP2 → single misprediction at the end of the loop
LOOP → 100% success rate

LOOP2 → at most 3 initial misprediction (2 at the begin and one at the end), then single misprediction
LOOP → 100% success rate

2bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LOOP2
        SUBI R0 R0 2
        BNEZ R0 LOOP
```

R0 is set to 1
R1 is set to 300



LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

LOOP

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do not collide

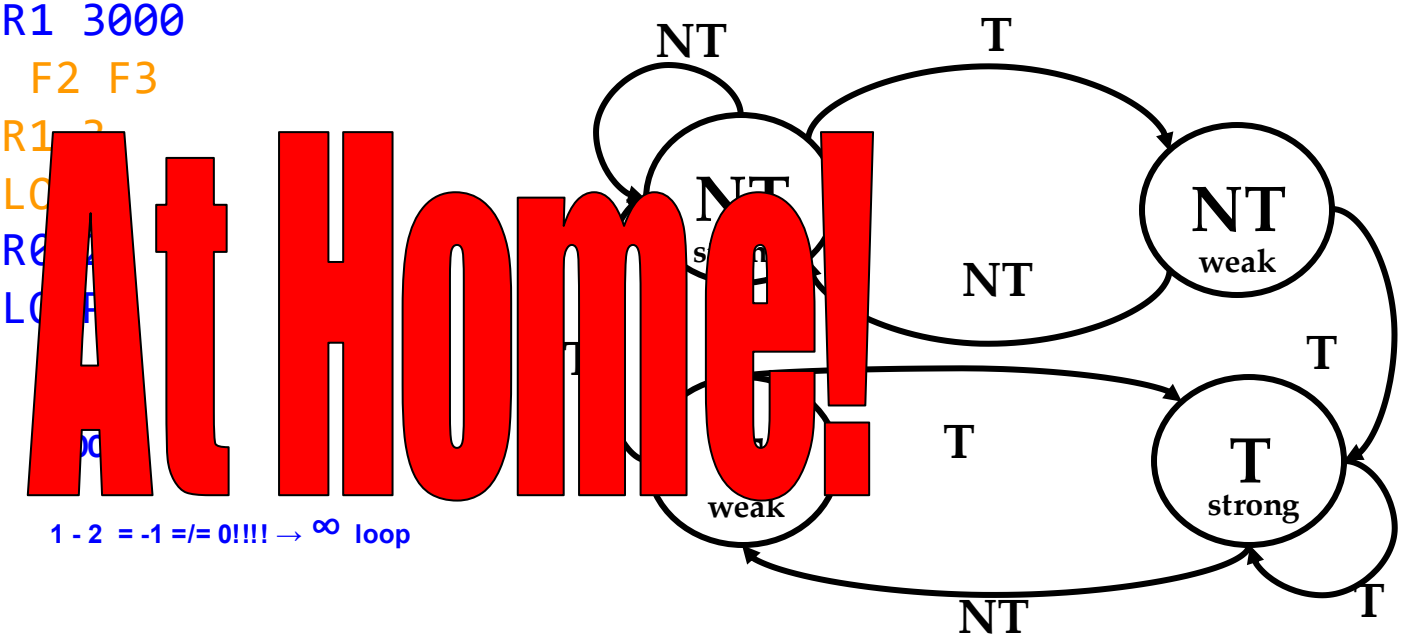
	2-BHT	2-BHT	2-BHT	2-BHT
LOOP:	T_{strong}	T	NT	NT
LOOP2:	T_{strong}	NT	T	NT

2bit - BHT

```
LOOP:   LD F3 0 (R0)
        ADDD F1 F3 F3
        ADDI R1 R1 3000
LOOP2:  MULTD F2 F2 F3
        SUBI R1 R1 3
        BNEZ R1 LO
        SUBI R0 R0
        BNEZ R0 LO
```

R0 is set to 1
R1 is set to 300

At Home!



LOOP2
@T0 → 3300 / 3 = 1100

@Ti 3000 / 3 = 1000 iterations

1 - 2 = -1 =/= 0!!!! → ∞ loop

Let us consider that the branch addresses do not collide

	2-BHT	2-BHT	2-BHT	2-BHT
LOOP:	T_{strong}	T_{strong}	…	…
LOOP2:	T_{strong}	T_{weak}	…	…

SUMMARY

Assumption: NO collision

WORST CASES

1-BHT

LOOP:	NT
LOOP2:	NT

2*∞ misprediction for LOOP2
1 misprediction for LOOP.

2-BHT

LOOP:	NT_{strong}
LOOP2:	NT_{strong}

2+1@T0 +1*∞ misprediction for LOOP2
2@T0 misprediction for LOOP.

BEST CASES

1-BHT

LOOP:	T
LOOP2:	T

1@T0 + 2*∞ misprediction for LOOP2
0 for LOOP

2-BHT

LOOP:	T_{strong}
LOOP2:	T_{strong}

1*∞ misprediction for LOOP2
0 for LOOP

SUMMARY

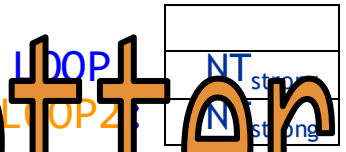
Assumption: NO collision

WORST CASES

1-BHT

2*∞ misprediction for LOOP2
1 misprediction for LOOP.

2-BHT



2+1@T0 +1*∞ misprediction for LOOP2
2@T0 misprediction for LOOP.

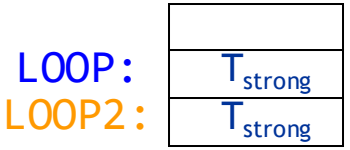
WORST 2BHT better than BEST 1BHT

BEST CASES

1-BHT

1@T0 + 2*∞ misprediction for LOOP2
0 for LOOP

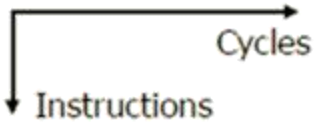
2-BHT

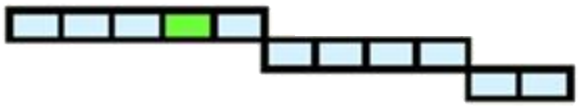


1*∞ misprediction for LOOP2
0 for LOOP

Recall: The ILP Architecture Journey

Steps towards exploiting more ILP

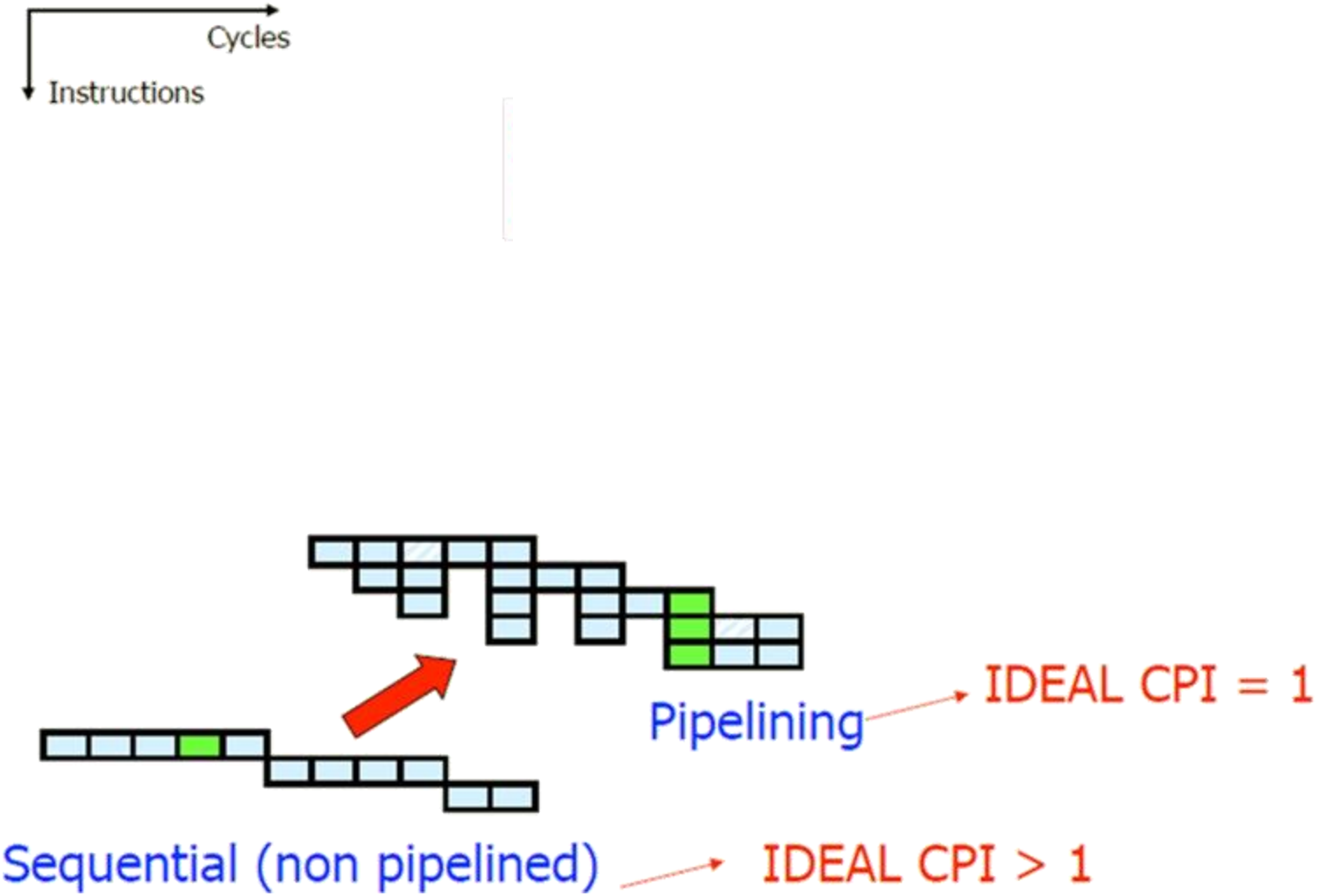




Sequential (non pipelined) → IDEAL CPI > 1

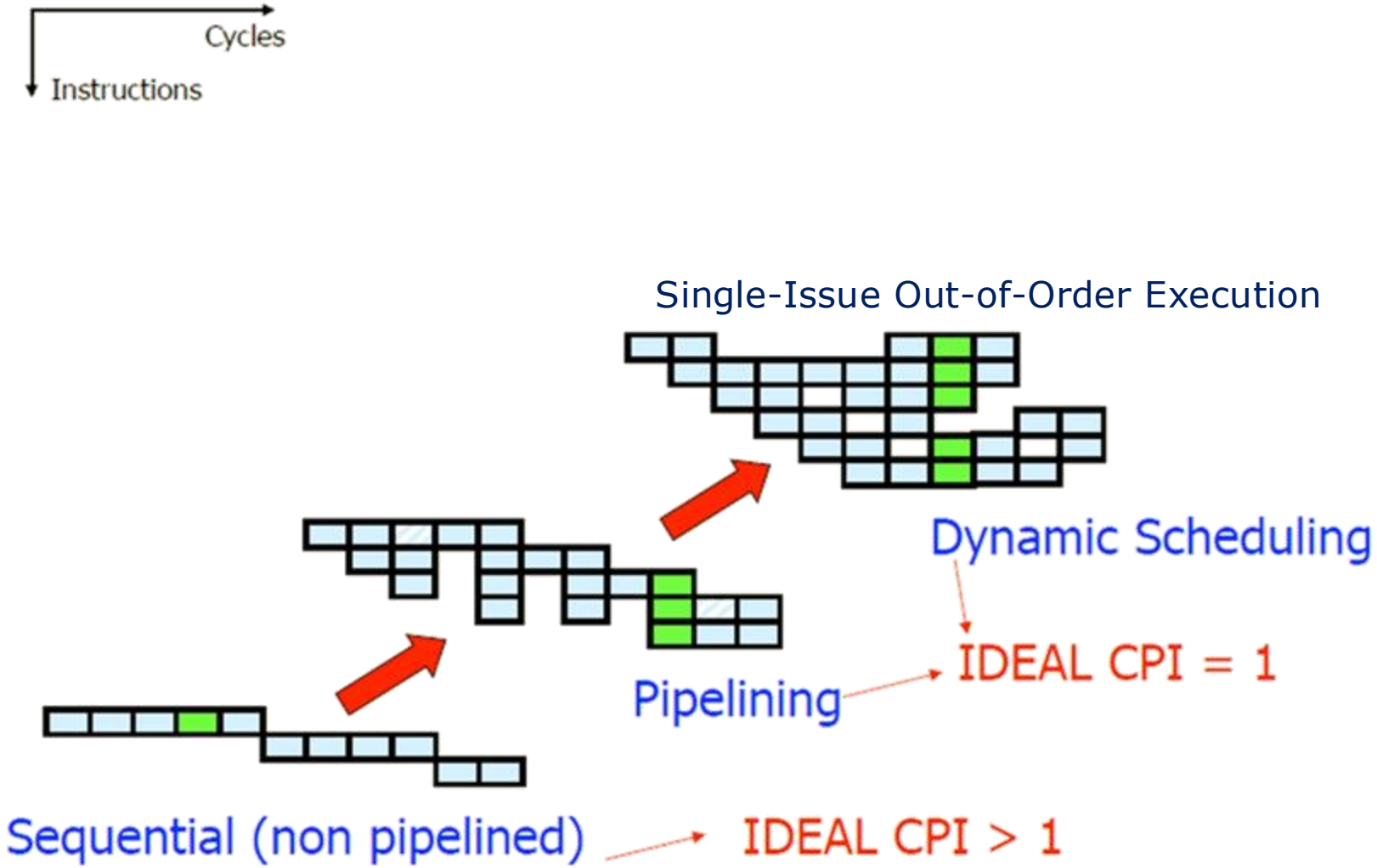
Recall: The ILP Architecture Journey

Steps towards exploiting more ILP



Recall: The ILP Architecture Journey

Steps towards exploiting more ILP



Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Solution: allow instructions behind a stall to proceed

- HW rearranges the instruction execution to reduce stalls

Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Solution: allow instructions behind a stall to proceed

- HW rearranges the instruction execution to reduce stalls

Enables out-of-order execution and completion (commit)

- Out-of order execution introduces possibility of WAR, WAW data hazards.

Recall: Key Idea: dynamic scheduling

Problem:

data dependences that cannot be hidden with bypassing or forwarding cause hardware stalls of the pipeline

Solution: allow instructions behind a stall to proceed

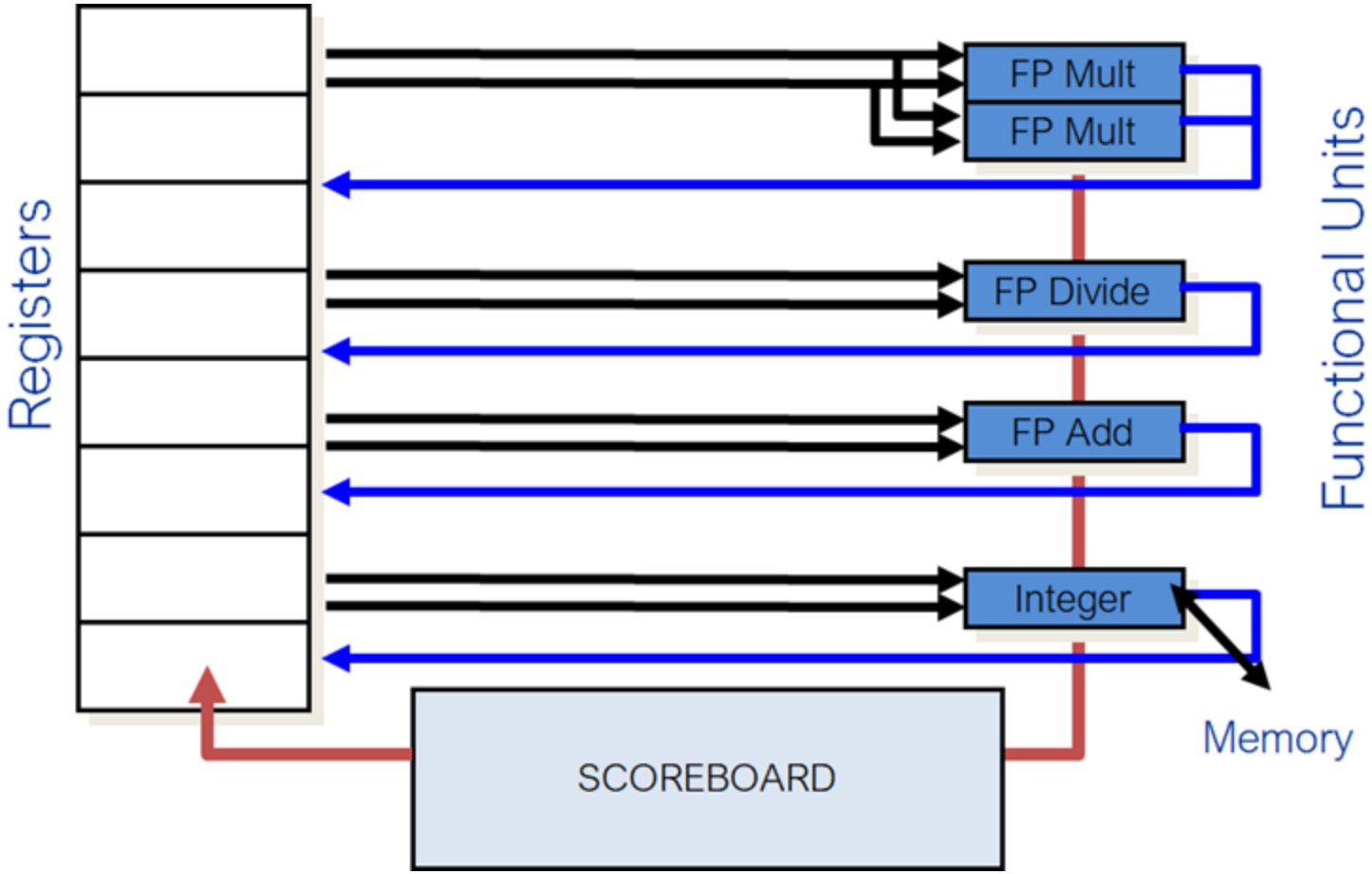
- HW rearranges the instruction execution to reduce stalls

Enables out-of-order execution and completion (commit)

- Out-of order execution introduces possibility of WAR, WAW data hazards.

First implemented in CDC6600 (1963)

Exe 1 Scoreboard



[Parallel operation in the control data 6600]

Recall: the Scoreboard pipeline

ISSUE	READ OPERAND	EXE COMPLETE	WB
Decode instruction;	Read operands;	Operate on operands;	Finish exec;
Structural FUs check; WAW checks	RAW check;	Notify Scoreboard on completion;	WAR &Struct check (FUs will hold results); Can overlap issue/read&write 4 Structural Hazard;

Exe 1 Scoreboard: the Code

```
I1:  LD F6 32+ R2
I2:  ADDD F2 F6 F4
I3:  MULTD F0 F4 F2
I4:  SUBD F12 F2 F6
I5:  ADDD F0 F12 F2
```

Exe 1.1 Scoreboard: Conflicts

```
I1:  LD F6 32+ R2
I2:  ADDD F2 F6 F4
I3:  MULTD F0 F4 F2
I4:  SUBD F12 F2 F6
I5:  ADDD F0 F12 F2
```

Exe 1.1 Scoreboard: Conflicts

RAW F6 I1-I2

```
I1:   LD   F6  32+ R2
I2:   ADDD F2  F6  F4
I3:   MULTD F0 F4 F2
I4:   SUBD F12 F2 F6
I5:   ADDD F0 F12 F2
```

Exe 1.1 Scoreboard: Conflicts

```
I1:   LD   F6  32+ R2
I2:   ADDD F2  F6  F4
I3:   MULTD F0 F4  F2
I4:   SUBD F12 F2  F6
I5:   ADDD F0 F12 F2
```

RAW F6 I1-I2

RAW F6 I1-I4

Exe 1.1 Scoreboard: Conflicts

```
I1:   LD   F6  32+ R2
I2:   ADDD F2  F6  F4
I3:   MULTD F0 F4  F2
I4:   SUBD F12 F2  F6
I5:   ADDD F0 F12 F2
```

RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

Exe 1.1 Scoreboard: Conflicts

```
I1:   LD   F6  32+ R2
I2:   ADDD F2  F6  F4
I3:   MULTD F0 F4  F2
I4:   SUBD F12 F2  F6
I5:   ADDD F0 F12 F2
```

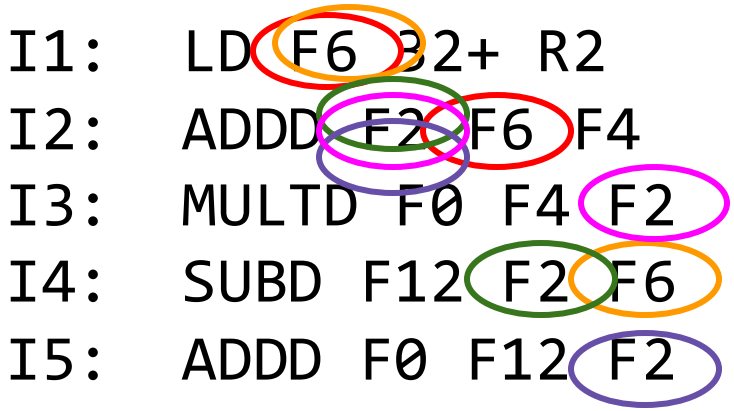
RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

Exe 1.1 Scoreboard: Conflicts



RAW F6 I1-I2

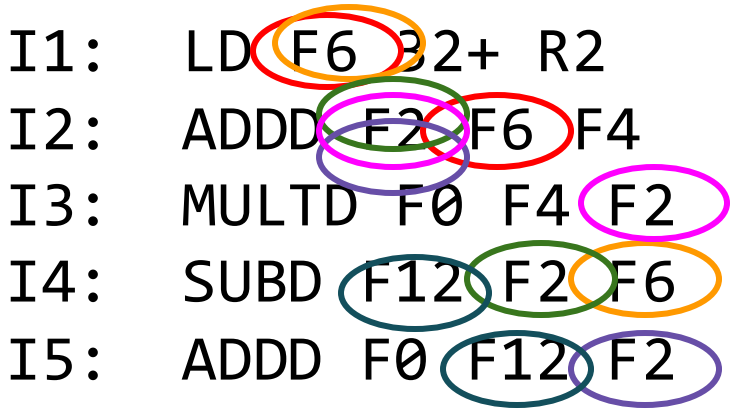
RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

RAW F2 I2-I5

Exe 1.1 Scoreboard: Conflicts



RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

RAW F2 I2-I5

RAW F12 I4-I5

Exe 1.1 Scoreboard: Conflicts

RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

RAW F2 I2-I5

RAW F12 I4-I5

WAW F0 I3-I5

Exe 1.2 Scoreboard: ∃ a configuration?

	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

Exe 1.2 Scoreboard: ∃ a configuration?

	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

Exe 1.2 Scoreboard: ∃ a configuration?

	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

Exe 1.2 Scoreboard: ∃ a configuration?

	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

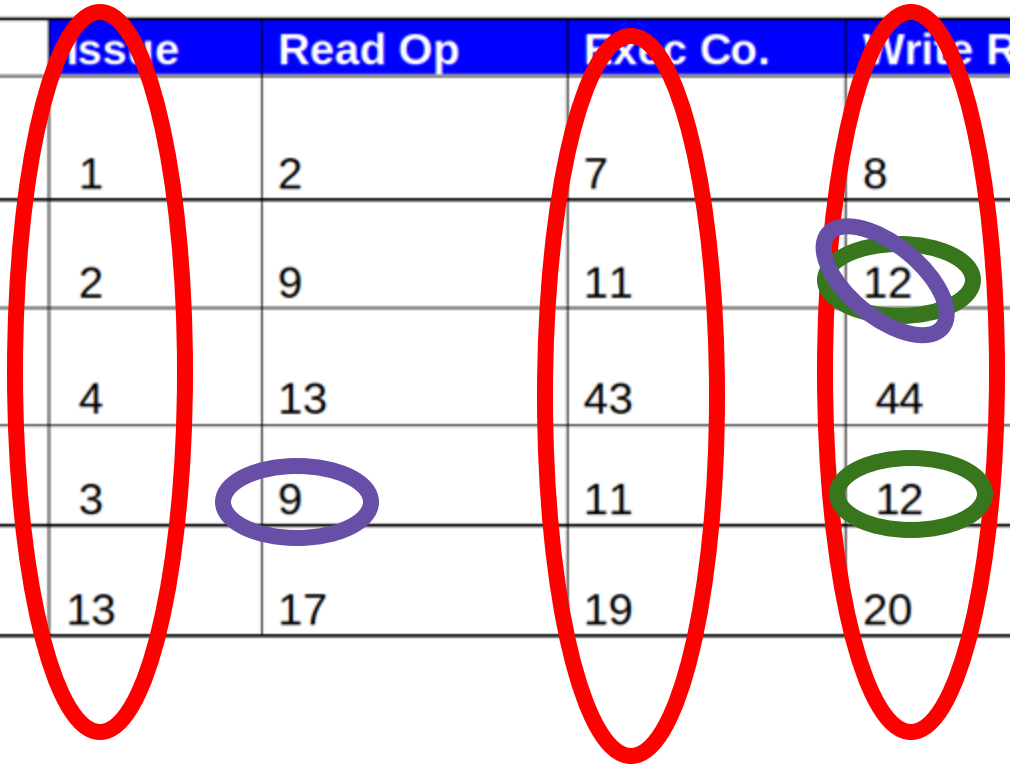
Exe 1.2 Scoreboard: ∃ a configuration?

	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20

- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

Exe 1.2 Scoreboard: ∃ a configuration?

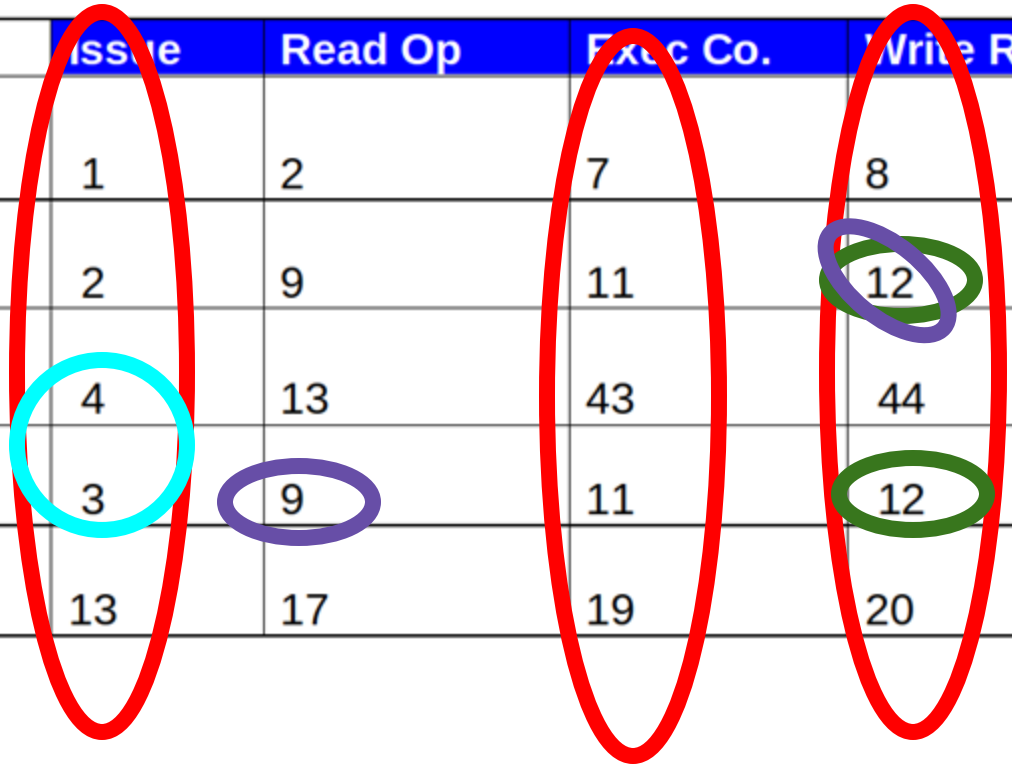
	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20



- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

Exe 1.2 Scoreboard: ∃ a configuration?

	Issue	Read Op	Exec Co.	Write R.
I1: LD F6 32+ R2	1	2	7	8
I2: ADDD F2 F6 F4	2	9	11	12
I3: MULTD F0 F4 F2	4	13	43	44
I4: SUBD F12 F2 F6	3	9	11	12
I5: ADDD F0 F12 F2	13	17	19	20



- Is there a “configuration” that can respect the shown execution?
- How many units? Which kind? What latency?

Exe 1.3 Scoreboard: if not correct, write right one

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2						
I2	ADDD F2 F6 F4						
I3	MULTD F0 F4 F2						
I4	SUBD F12 F2 F6						
I5	ADDD F0 F12 F2						

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

Exe 1.3 Scoreboard: if not correct, write right one CC 0

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2						
I2	ADDD F2 F6 F4						
I3	MULTD F0 F4 F2						
I4	SUBD F12 F2 F6						
I5	ADDD F0 F12 F2						

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F6 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 1

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1					MU
I2	ADDD F2 F6 F4						
I3	MULTD F0 F4 F2						
I4	SUBD F12 F2 F6						
I5	ADDD F0 F12 F2						

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F6 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 2

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2				MU
I2	ADDD F2 F6 F4	2					FPU1
I3	MULTD F0 F4 F2						
I4	SUBD F12 F2 F6						
I5	ADDD F0 F12 F2						

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F6 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 3

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2				MU
I2	ADDD F2 F6 F4	2				RAW F6	FPU1
I3	MULTD F0 F4 F2	3					FPU2
I4	SUBD F12 F2 F6						
I5	ADDD F0 F12 F2						

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F0 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 4

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4			MU
I2	ADDD F2 F6 F4	2				RAW F6	FPU1
I3	MULTD F0 F4 F2	3				RAW F2	FPU2
I4	SUBD F12 F2 F6	4					FPU3
I5	ADDD F0 F12 F2						

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 5

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2				RAW F6	FPU1
I3	MULTD F0 F4 F2	3				RAW F2	FPU2
I4	SUBD F12 F2 F6	4					FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F6 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 5

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2				RAW F6	FPU1
I3	MULTD F0 F4 F2	3				RAW F2	FPU2
I4	SUBD F12 F2 F6	4				RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

RAW F6 I1-I2
RAW F6 I1-I4
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 6

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6			RAW F6	FPU1
I3	MULTD F0 F4 F2	3				RAW F2	FPU2
I4	SUBD F12 F2 F6	4				RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 9

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9		RAW F6	FPU1
I3	MULTD F0 F4 F2	3				RAW F2	FPU2
I4	SUBD F12 F2 F6	4				RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
RAW F2 I2-I3
RAW F2 I2-I4
RAW F2 I2-I5
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 10

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3				RAW F2	FPU2
I4	SUBD F12 F2 F6	4				RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 11

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11			RAW F2	FPU2
I4	SUBD F12 F2 F6	4	11			RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 14

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11	14		RAW F2	FPU2
I4	SUBD F12 F2 F6	4	11	14		RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 14

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11	14		RAW F4	FPU2
I4	SUBD F12 F2 F6	4	11	14		RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

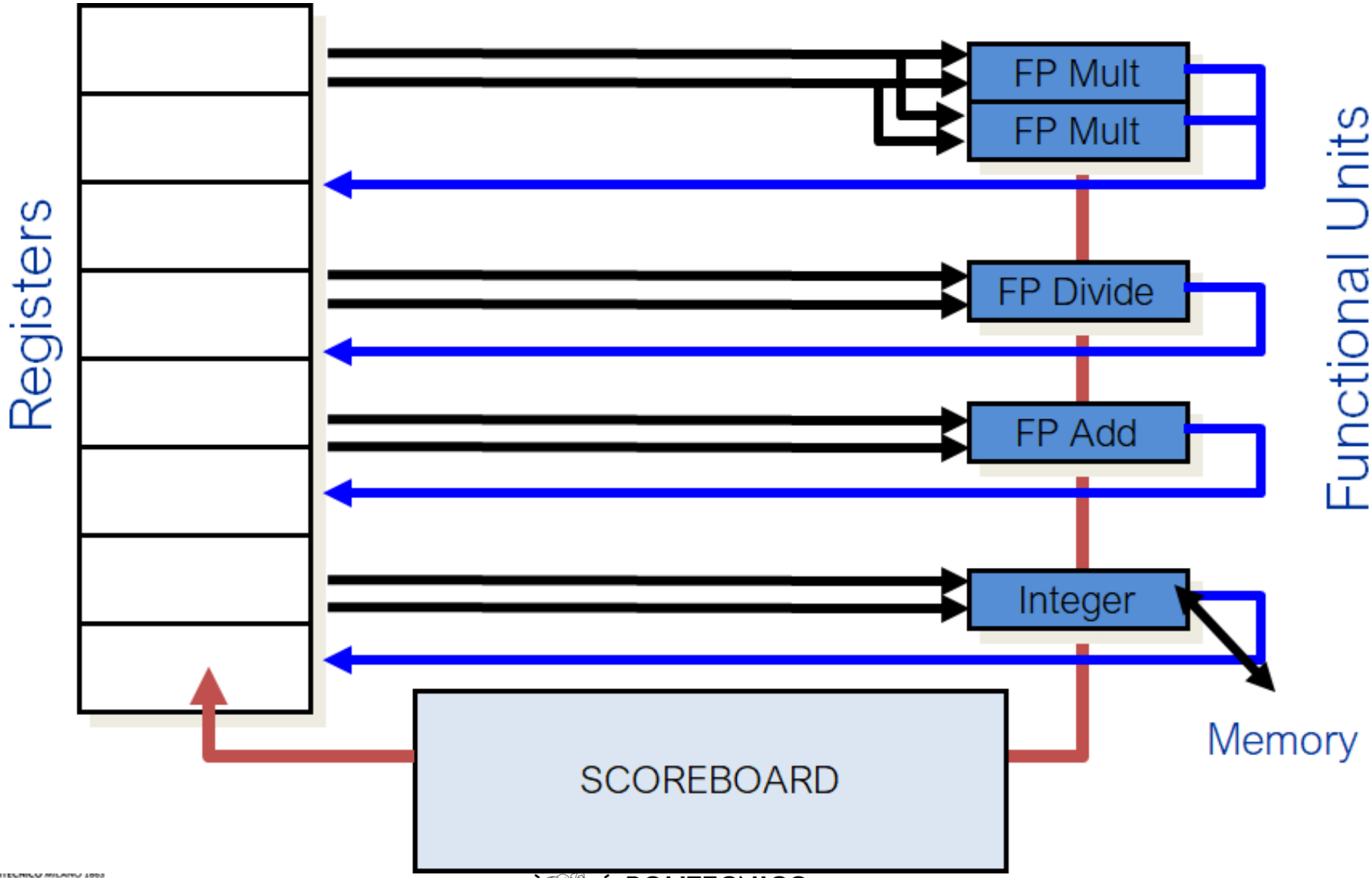
SINGLE WRITE PORT?!?!?!?!?!?!?

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one



Exe 1.3 Scoreboard: if not correct, write right one CC 15

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11	14	15	RAW F2	FPU2
I4	SUBD F12 F2 F6	4	11	14		RAW F2 + Struct RF	FPU3
I5	ADDD F0 F12 F2					WAW F0	

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, <u>single write</u> port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
RAW F12 I4-I5
WAW F0 I3-I5

Exe 1.3 Scoreboard: if not correct, write right one CC 16

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11	14	15	RAW F2	FPU2
I4	SUBD F12 F2 F6	4	11	14	16	RAW F2 + Struct RF	FPU3
I5	ADDD F0 F12 F2	16				WAW F0	FPU4

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
~~RAW F12 I4-I5~~
~~WAW F0 I3-I5~~

Exe 1.3 Scoreboard: if not correct, write right one CC 17

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11	14	15	RAW F2	FPU2
I4	SUBD F12 F2 F6	4	11	14	16	RAW F2 + Struct RF	FPU3
I5	ADDD F0 F12 F2	16	17			WAW F0	FPU4

If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, <u>single write</u> port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
~~RAW F12 I4-I5~~
~~WAW F0 I3-I5~~

Exe 1.3 Scoreboard: if not correct, write right one CC 21

	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB	Hazards	Unit
I1	LD F6 32+ R2	1	2	4	5		MU
I2	ADDD F2 F6 F4	2	6	9	10	RAW F6	FPU1
I3	MULTD F0 F4 F2	3	11	14	15	RAW F2	FPU2
I4	SUBD F12 F2 F6	4	11	14	16	RAW F2 + Struct RF	FPU3
I5	ADDD F0 F12 F2	16	17	20	21	WAW F0	FPU4

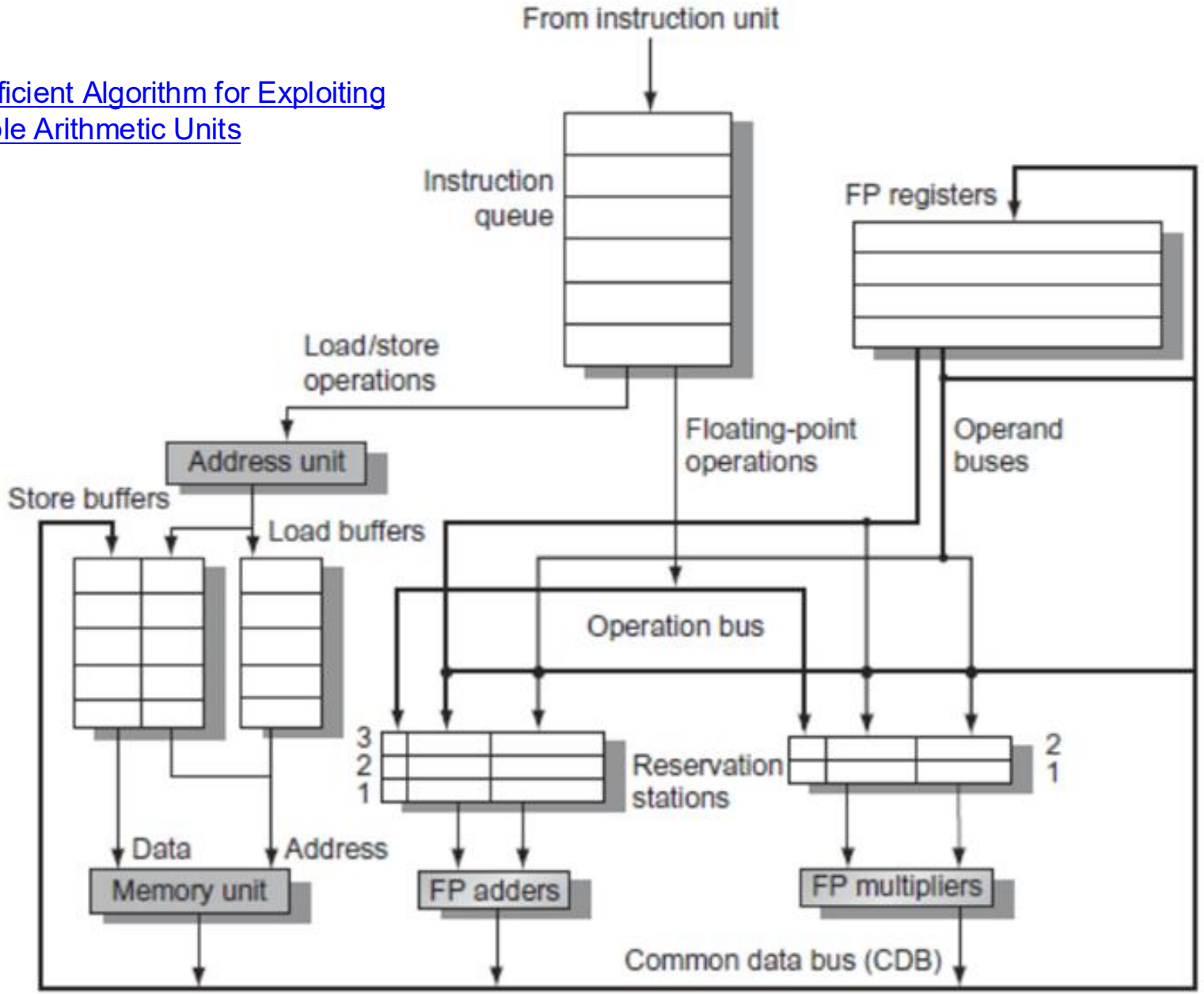
If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

4 FPALU 3 cc latency, single write port for the pool
1 MEM 2 cc latency

~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2-I4~~
~~RAW F2 I2-I5~~
~~RAW F12 I4-I5~~
~~WAW F0 I3-I5~~

Exe Tomasulo

[An Efficient Algorithm for Exploiting Multiple Arithmetic Units]



Recall: the Tomasulo pipeline

ISSUE	EXECUTION	WRITE
Get Instruction from Queue and Rename Registers	Execute and Watch CDB;	Write on CDB;
Structural RSs check; WAW and WAR solved by Renaming (!!!in-order-issue!!!);	Check for Struct on FUs; RAW delaying; Struct check on CDB;	(FUs will hold results unless CDB free) RSs/FUs marked free

Exe .1 Tomasulo: Code

```
I1:  lw $f1, 0($r0)
I2:  faddi $f1, $f1, C1
I3:  faddi $f2, $f1, C2
I4:  sw $f2, 0($r0)
I5:  lw $f2, 4($r0)
I6:  fadd $f2, $f2, $f2
I7:  sw $f2, 4($r0)
```

Exe .1 Tomasulo: Conflicts

```
I1:   lw $f1, 0($r0)
I2:   faddi $f1, $f1, C1
I3:   faddi $f2, $f1, C2
I4:   sw $f2, 0($r0)
I5:   lw $f2, 4($r0)
I6:   fadd $f2, $f2, $f2
I7:   sw $f2, 4($r0)
```

RAW f1 I1-I2
RAW f1 I2-I3
RAW f2 I3-I4
RAW f2 I5-I6
RAW f2 I6-I7

WAW f2 I5-I6
WAW f2 I5-I3
WAW f1 I2-I1
WAW f2 I6-I3
WAR f2 I4-I5
WAR f2 I4-I6

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`						
I2: `faddi $f1, $f1, C1`						
I3: `faddi $f2, $f1, C2`						
I4: `sw $f2, 0($r0)`						
I5: `lw $f2, 4($r0)`						
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 **RAW f1 I2-I3** **RAW f2 I3-I4** **RAW f2 I5-I6** **RAW f2 I6-I7**

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1				RS1	LDU1
I2: `faddi $f1, $f1, C1`						
I3: `faddi $f2, $f1, C2`						
I4: `sw $f2, 0($r0)`						
I5: `lw $f2, 4($r0)`						
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 **RAW f1 I2-I3** **RAW f2 I3-I4** **RAW f2 I5-I6** **RAW f2 I6-I7**

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2			RS1	LDU1
I2: `faddi $f1, $f1, C1`	2			RAW $f1	RS3	
I3: `faddi $f2, $f1, C2`						
I4: `sw $f2, 0($r0)`						
I5: `lw $f2, 4($r0)`						
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2			RS1	LDU1
I2: `faddi $f1, $f1, C1`	2			RAW $f1	RS3	
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`						
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo: CC5?

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2			RS1	LDU1
I2: `faddi $f1, $f1, C1`	2			RAW $f1	RS3	
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`						
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo: CC5?

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2			RS1	LDU1
I2: `faddi $f1, $f1, C1`	2			RAW $f1	RS3	
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`				struct RS1		
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2			RAW $f1	RS3	
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`				struct RS1		
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7		RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`	7			struct RS1	RS1	
I6: `fadd $f2, $f2, $f2`						
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo: CC8?

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7		RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`				struct RS3		
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3			RAW $f1 (struct ALU1)	RS4	
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`				struct RS3		
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10		RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`	10			struct RS3	RS3	
I7: `sw $f2, 4($r0)`						

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo: CC11?

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10		RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`	10			struct RS3	RS3	
I7: `sw $f2, 4($r0)`				struct RS2		

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4			RAW $f2 (struct LDU1)	RS2	
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`	10			struct RS3 + RAW $f2 (struct ALU1)	RS3	
I7: `sw $f2, 4($r0)`				struct RS2		

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13		RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`	10			struct RS3 + RAW $f2 (struct ALU1)	RS3	
I7: `sw $f2, 4($r0)`				struct RS2		

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7			struct RS1 + struct LDU1	RS1	
I6: `fadd $f2, $f2, $f2`	10			struct RS3 + RAW $f2 (struct ALU1)	RS3	
I7: `sw $f2, 4($r0)`				struct RS2		

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18		struct RS1 + struct LDU1	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10			struct RS3 + RAW $f2 (struct ALU1)	RS3	
I7: `sw $f2, 4($r0)`	18			struct RS2	RS2	

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo: CC19?

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18		struct RS1 + struct LDU1	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10			struct RS3 + RAW $f2 (struct ALU1)	RS3	
I7: `sw $f2, 4($r0)`	18			struct RS2 + RAW $f2 (struct LDU1)	RS2	

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	**RAW $f1**	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	**RAW $f1** (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	**RAW $f2** (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18	22	**struct RS1 + struct LDU1**	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10			**struct RS3 + RAW $f2** (struct ALU1)	RS3	
I7: `sw $f2, 4($r0)`	18			**struct RS2 + RAW $f2** (struct LDU1)	RS2	

RAW f1 I1-I2 **RAW f1 I2-I3** **RAW f2 I3-I4** **RAW f2 I5-I6** **RAW f2 I6-I7**

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18	22	struct RS1 + struct LDU1	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10	23		struct RS3 + RAW $f2 (struct ALU1)	RS3	ALU1
I7: `sw $f2, 4($r0)`	18			struct RS2 + RAW $f2 (struct LDU1)	RS2	

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18	22	struct RS1 + struct LDU1	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10	23	25	struct RS3 + RAW $f2 (struct ALU1)	RS3	ALU1
I7: `sw $f2, 4($r0)`	18			struct RS2 + RAW $f2 (struct LDU1)	RS2	

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18	22	struct RS1 + struct LDU1	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10	23	25	struct RS3 + RAW $f2 (struct ALU1)	RS3	ALU1
I7: `sw $f2, 4($r0)`	18	26		struct RS2 + RAW $f2 (struct LDU1)	RS2	LDU1

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Exe 3.2 Tomasulo

- **2** RESERVATION STATIONS (**RS1, RS2**) + **1** LOAD/STORE unit (**LDU1**) with latency **4**
- **2** RESERVATION STATIONS (**RS3, RS4**) + **1** ALU/BR FUs (**ALU1**) with latency **2**

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: `lw $f1, 0($r0)`	1	2	6		RS1	LDU1
I2: `faddi $f1, $f1, C1`	2	7	9	RAW $f1	RS3	ALU1
I3: `faddi $f2, $f1, C2`	3	10	12	RAW $f1 (struct ALU1)	RS4	ALU1
I4: `sw $f2, 0($r0)`	4	13	17	RAW $f2 (struct LDU1)	RS2	LDU1
I5: `lw $f2, 4($r0)`	7	18	22	struct RS1 + struct LDU1	RS1	LDU1
I6: `fadd $f2, $f2, $f2`	10	23	25	struct RS3 + RAW $f2 (struct ALU1)	RS3	ALU1
I7: `sw $f2, 4($r0)`	18	26	30	struct RS2 + RAW $f2 (struct LDU1)	RS2	LDU1

RAW f1 I1-I2 RAW f1 I2-I3 RAW f2 I3-I4 RAW f2 I5-I6 RAW f2 I6-I7

Thank you for your attention
Questions?

Alessandro Verosimile <alessandro.verosimile@polimi.it>

Acknowledgements

Davide Conficconi, E. Del Sozzo, Marco D. Santambrogio, D. Sciuto

Part of this material comes from:

- "Computer Organization and Design" and "Computer Architecture A Quantitative Approach" Patterson and Hennessy books
- News and paper cited throughout the lecture