Exercise Session 8

Tomasulo, Dyamic Branch Predictor, VLIW

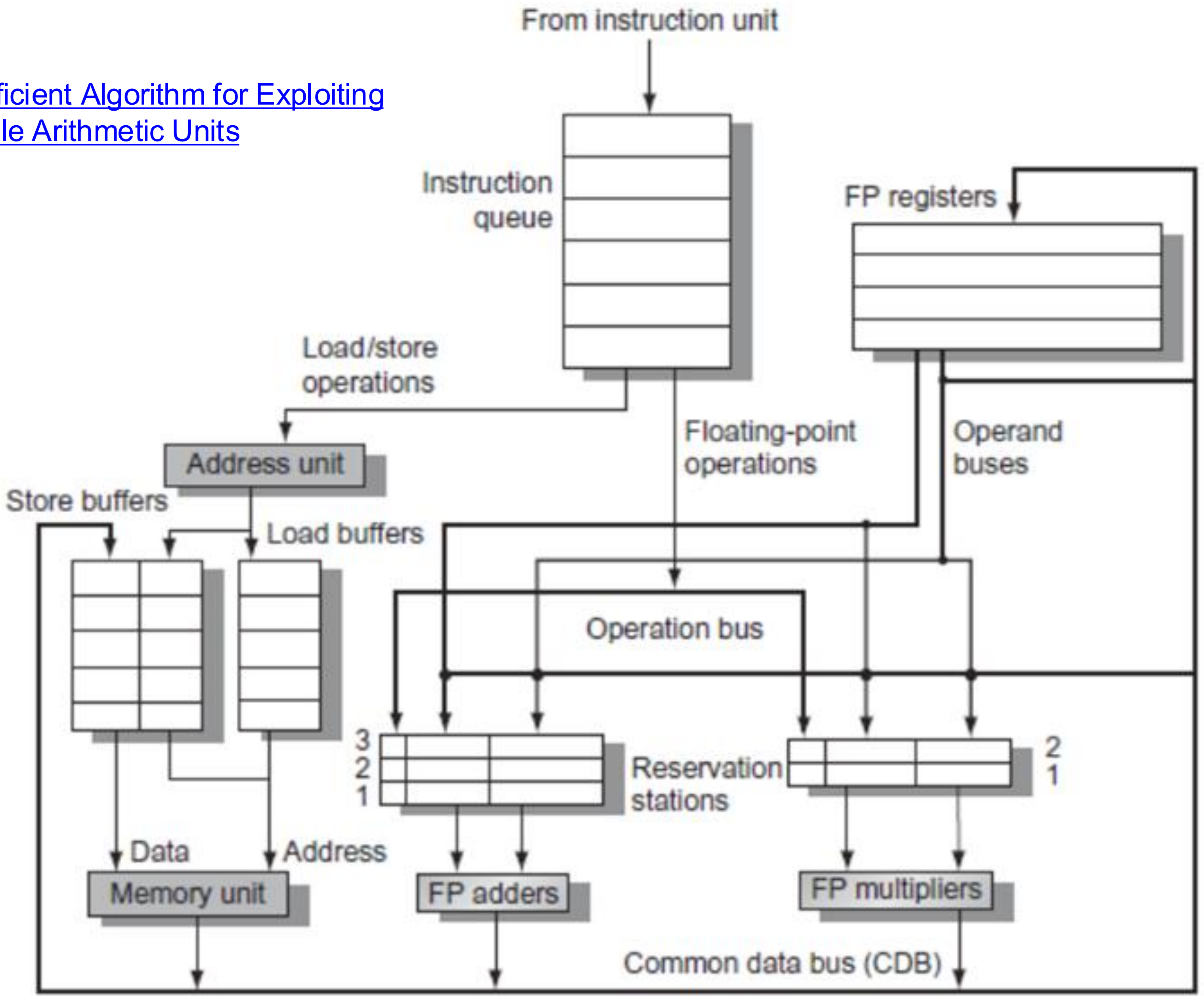
Advanced Computer Architectures

Politecnico di Milano
May 26th, 2025

Alessandro Verosimile <alessandro.verosimile@polimi.it>

Exe.3 Tomasulo

An Efficient Algorithm for Exploiting Multiple Arithmetic Units



Exe.3 Tomasulo

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Exe.3 The conflicts

```
I1:  LD $F1, 0($R1)
I2:  FADD $F2, $F2, $F3
I3:  ADDI $R3, $R3, 8
I4:  LD $F4, 0(R2)
I5:  FADD $F5, $F4, $F2
I6:  FMULT $F6, $F1, $F4
I7:  ADDI $R5, $R5, 1
I8:  LD $R6, 0($R4)
I9:  SD $F6, 0($R5)
I10: SD $F5, 0($R6)
```

Exe.3 The conflicts

```
I1:  LD $F1, 0($R1)
I2:  FADD $F2, $F2, $F3
I3:  ADDI $R3, $R3, 8
I4:  LD $F4, 0(R2)
I5:  FADD $F5, $F4, $F2
I6:  FMULT $F6, $F1, $F4
I7:  ADDI $R5, $R5, 1
I8:  LD $R6, 0($R4)
I9:  SD $F6, 0($R5)
I10: SD $F5, 0($R6)
```

RAW F0 I1–I6

RAW F2 I2–I5

RAW F4 I4–I5

RAW F4 I4–I6

RAW F5 I5–I10

RAW F6 I6–I9

RAW R5 I7–I9

RAW R6 I8–I10

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.3 Tomasulo CC0

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)						
I2: FADD $F2, $F2, $F3						
I3: ADDI $R3, $R3, 8						
I4: LD $F4, 0(R2)						
I5: FADD $F5, $F4, $F2						
I6: FMULT $F6, $F1, $F4						
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

RAW F0 I1-I6 **RAW F2 I2-I5** **RAW F4 I4-I6** **RAW F4 I4-I5**

RAW F5 I5-I10 **RAW F6 I6-I9** **RAW R5 I7-I9** **RAW R6 I8-I10**

Exe.3 Tomasulo CC1

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	**1**				RS1	
I2: FADD $F2, $F2, $F3						
I3: ADDI $R3, $R3, 8						
I4: LD $F4, 0(R2)						
I5: FADD $F5, $F4, $F2						
I6: FMULT $F6, $F1, $F4						
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

RAW F0 I1-I6 **RAW F2 I2-I5** **RAW F4 I4-I6** **RAW F4 I4-I5**

RAW F5 I5-I10 **RAW F6 I6-I9** **RAW R5 I7-I9** **RAW R6 I8-I10**

Exe.3 Tomasulo CC2

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	**1**	**2**			RS1	LDU1
I2: FADD $F2, $F2, $F3	**2**				RS4	
I3: ADDI $R3, $R3, 8						
I4: LD $F4, 0(R2)						
I5: FADD $F5, $F4, $F2						
I6: FMULT $F6, $F1, $F4						
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

RAW F0 I1-I6 **RAW F2 I2-I5** **RAW F4 I4-I6** **RAW F4 I4-I5**

RAW F5 I5-I10 **RAW F6 I6-I9** **RAW R5 I7-I9** **RAW R6 I8-I10**

Exe.3 Tomasulo CC3

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	**1**	**2**			RS1	LDU1
I2: FADD $F2, $F2, $F3	**2**	**3**			RS4	FPU1
I3: ADDI $R3, $R3, 8	**3**				RS7	
I4: LD $F4, 0(R2)						
I5: FADD $F5, $F4, $F2						
I6: FMULT $F6, $F1, $F4						
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

RAW F0 I1-I6 **RAW F2 I2-I5** **RAW F4 I4-I6** **RAW F4 I4-I5**

RAW F5 I5-I10 **RAW F6 I6-I9** **RAW R5 I7-I9** **RAW R6 I8-I10**

Exe.3 Tomasulo CC4

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	**1**	**2**			RS1	LDU1
I2: FADD $F2, $F2, $F3	**2**	**3**			RS4	FPU1
I3: ADDI $R3, $R3, 8	**3**	**4**			RS7	ALU1
I4: LD $F4, 0(R2)	**4**				RS2	
I5: FADD $F5, $F4, $F2						
I6: FMULT $F6, $F1, $F4						
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

RAW F0 I1-I6 **RAW F2 I2-I5** **RAW F4 I4-I6** **RAW F4 I4-I5**

RAW F5 I5-I10 **RAW F6 I6-I9** **RAW R5 I7-I9** **RAW R6 I8-I10**

Exe.3 Tomasulo CC5

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	**1**	**2**	**5**		RS1	LDU1
I2: FADD $F2, $F2, $F3	**2**	**3**			RS4	FPU1
I3: ADDI $R3, $R3, 8	**3**	**4**	○	**Struct CDB**	RS7	ALU1
I4: LD $F4, 0(R2)	**4**	**5**			RS2	LDU2
I5: FADD $F5, $F4, $F2	**5**				RS5	
I6: FMULT $F6, $F1, $F4						
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

RAW F0 I1-I6 **RAW F2 I2-I5** **RAW F4 I4-I6** **RAW F4 I4-I5**

RAW F5 I5-I10 **RAW F6 I6-I9** **RAW R5 I7-I9** **RAW R6 I8-I10**

Exe.3 Tomasulo CC6

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4		Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5			RS2	LDU2
I5: FADD $F5, $F4, $F2	5			RAW $F4, RAW $F2	RS5	
I6: FMULT $F6, $F1, $F4	6				RS6	
I7: ADDI $R5, $R5, 1						
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

~~RAW F0 I1-I6~~ RAW F2 I2-I5 RAW F4 I4-I6 RAW F4 I4-I5

RAW F5 I5-I10 RAW F6 I6-I9 RAW R5 I7-I9 RAW R6 I8-I10

Exe.3 Tomasulo CC7

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5			RS2	LDU2
I5: FADD $F5, $F4, $F2	5			RAW $F4, RAW $F2	RS5	
I6: FMULT $F6, $F1, $F4	6			RAW $F4	RS6	
I7: ADDI $R5, $R5, 1	7				RS8	
I8: LD $R6, 0($R4)						
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ RAW F4 I4-I6 RAW F4 I4-I5

RAW F5 I5-I10 RAW F6 I6-I9 RAW R5 I7-I9 RAW R6 I8-I10

Exe.3 Tomasulo CC8

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5			RAW $F4, RAW $F2	RS5	
I6: FMULT $F6, $F1, $F4	6			RAW $F4	RS6	
I7: ADDI $R5, $R5, 1	7	8			RS8	ALU1
I8: LD $R6, 0($R4)	8				RS1	
I9: SD $F6, 0($R5)						
I10: SD $F5, 0($R6)						

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ RAW F4 I4-I6 RAW F4 I4-I5

RAW F5 I5-I10 RAW F6 I6-I9 RAW R5 I7-I9 RAW R6 I8-I10

Exe.3 Tomasulo CC9

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9		RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9		RAW $F4	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9			RS1	LDU1
I9: SD $F6, 0($R5)	9				RS2	
I10: SD $F5, 0($R6)						

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

RAW F5 I5-I10 RAW F6 I6-I9 RAW R5 I7-I9 RAW R6 I8-I10

Exe.3 Tomasulo CC10

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9		RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9		RAW $F4	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9			RS1	LDU1
I9: SD $F6, 0($R5)	9			RAW $F6	RS2	
I10: SD $F5, 0($R6)	10				RS3	

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

RAW F5 I5-I10 RAW F6 I6-I9 ~~RAW R5 I7-I9~~ RAW R6 I8-I10

Exe.3 Tomasulo CC11

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD \$F1, 0(\$R1)	1	2	5		RS1	LDU1
I2: FADD \$F2, \$F2, \$F3	2	3	6		RS4	FPU1
I3: ADDI \$R3, \$R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD \$F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD \$F5, \$F4, \$F2	5	9		RAW \$F4, RAW \$F2	RS5	FPU1
I6: FMULT \$F6, \$F1, \$F4	6	9		RAW \$F4	RS6	FPU2
I7: ADDI \$R5, \$R5, 1	7	8	9		RS8	ALU1
I8: LD \$R6, 0(\$R4)	8	9			RS1	LDU1
I9: SD \$F6, 0(\$R5)	9			RAW \$F6	RS2	
I10: SD \$F5, 0(\$R6)	10			RAW \$F5, RAW \$R6	RS3	

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

RAW F5 I5-I10 RAW F6 I6-I9 ~~RAW R5 I7-I9~~ RAW R6 I8-I10

Exe.3 Tomasulo CC12

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9	12	RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9		RAW $F4, Struct CDB	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9		Struct CDB	RS1	LDU1
I9: SD $F6, 0($R5)	9			RAW $F6	RS2	
I10: SD $F5, 0($R6)	10			RAW $F5, RAW $R6	RS3	

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

RAW F5 I5-I10 RAW F6 I6-I19 ~~RAW R5 I7-I9~~ RAW R6 I8-I10

Exe.3 Tomasulo CC13

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9	12	RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9	13	RAW $F4, Struct CDB	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9		Struct CDB	RS1	LDU1
I9: SD $F6, 0($R5)	9			RAW $F6	RS2	
I10: SD $F5, 0($R6)	10			RAW $F5, RAW $R6	RS3	

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

~~RAW F5 I5-I10~~ RAW F6 I6-I9 ~~RAW R5 I7-I9~~ RAW R6 I8-I10

Exe.3 Tomasulo CC14

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9	12	RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9	13	RAW $F4, Struct CDB	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9	14	Struct CDB	RS1	LDU1
I9: SD $F6, 0($R5)	9	14		RAW $F6	RS2	LDU2
I10: SD $F5, 0($R6)	10			RAW $F5, RAW $R6	RS3	

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

~~RAW F5 I5-I10~~ ~~RAW F6 I6-I9~~ ~~RAW R5 I7-I9~~ RAW R6 I8-I10

Exe.3 Tomasulo CC15

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9	12	RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9	13	RAW $F4, Struct CDB	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9	14	Struct CDB	RS1	LDU1
I9: SD $F6, 0($R5)	9	14		RAW $F6	RS2	LDU2
I10: SD $F5, 0($R6)	10	15		RAW $F5, RAW $R6	RS3	LDU1

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

~~RAW F5 I5-I10~~ ~~RAW F6 I6-I9~~ ~~RAW R5 I7-I9~~ ~~RAW R6 I8-I10~~

Exe.3 Tomasulo CC17

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9	12	RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9	13	RAW $F4, Struct CDB	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9	14	Struct CDB	RS1	LDU1
I9: SD $F6, 0($R5)	9	14	17	RAW $F6	RS2	LDU2
I10: SD $F5, 0($R6)	10	15		RAW $F5, RAW $R6	RS3	LDU1

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

~~RAW F5 I5-I10~~ ~~RAW F6 I6-I9~~ ~~RAW R5 I7-I9~~ ~~RAW R6 I8-I10~~

Exe.3 Tomasulo CC18

- 3 RESERVATION STATIONS (RS1, RS2, RS3) + 3 LOAD/STORE unit (LDU1, LDU2, LDU3) with latency 3
- 3 RESERVATION STATIONS (RS4, RS5, RS6) + 3 FPUs (FPU1, FPU2, FPU3) with latency 3
- 2 RESERVATION STATIONS (RS7, RS8) + 1 Integer ALU (ALU1) with latency 1

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1: LD $F1, 0($R1)	1	2	5		RS1	LDU1
I2: FADD $F2, $F2, $F3	2	3	6		RS4	FPU1
I3: ADDI $R3, $R3, 8	3	4	7	Struct CDB	RS7	ALU1
I4: LD $F4, 0(R2)	4	5	8		RS2	LDU2
I5: FADD $F5, $F4, $F2	5	9	12	RAW $F4, RAW $F2	RS5	FPU1
I6: FMULT $F6, $F1, $F4	6	9	13	RAW $F4, Struct CDB	RS6	FPU2
I7: ADDI $R5, $R5, 1	7	8	9		RS8	ALU1
I8: LD $R6, 0($R4)	8	9	14	Struct CDB	RS1	LDU1
I9: SD $F6, 0($R5)	9	14	17	RAW $F6	RS2	LDU2
I10: SD $F5, 0($R6)	10	15	18	RAW $F5, RAW $R6	RS3	LDU1

~~RAW F0 I1-I6~~ ~~RAW F2 I2-I5~~ ~~RAW F4 I4-I6~~ ~~RAW F4 I4-I5~~

~~RAW F5 I5-I10~~ ~~RAW F6 I6-I9~~ ~~RAW R5 I7-I9~~ ~~RAW R6 I8-I10~~

Scoreboard vs Tomasulo

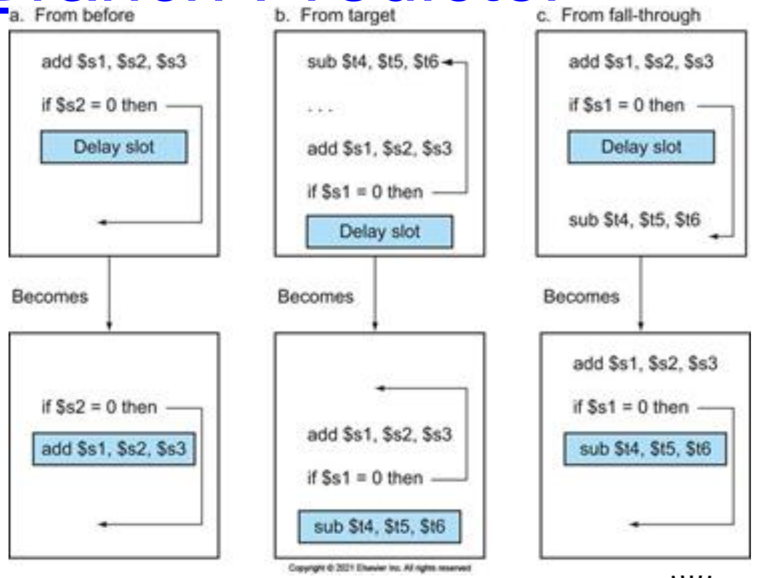
	Instruction	ISSUE	READ OPERAND	EXE COMPLETE	WB
I1	LD $F1, 0($R1)	1	2	5	6
I2	FADD $F2, $F2, $F3	2	3	7	8
I3	ADDI $R3, $R3, 8	3	4	5	7
I4	LD $F4, 0(R2)	4	5	8	9
I5	FADD $F5, $F4, $F2	5	10	14	15
I6	FMULT $F6, $F1, $F4	6	10	14	16
I7	ADDI $R5, $R5, 1	7	8	9	10
I8	LD $R6, 0($R4)	8	9	12	13
I9	SD $F6, 0($R5)	9	17	20	21
I10	SD $F5, 0($R6)	10	16	19	20

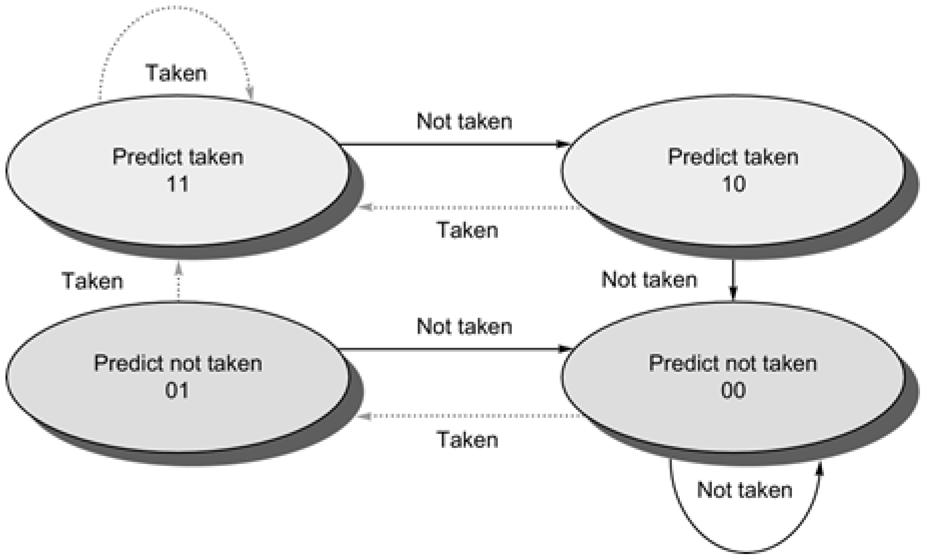
ISSUE	START EXE	WB
1	2	5
2	3	6
3	4	7
4	5	8
5	9	12
6	9	13
7	8	9
8	9	14
9	14	17
10	15	18

Prediction

[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 10, R1 is set to 0)

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 10
LOOP2:  LD F3 0 R1
        MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 10
        BNE R0 R1 LOOP
```

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

How many iterations?

R0 is set to 10
R1 is set to 0

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 10
LOOP2:  LD F3 0 R1
        MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 10
        BNEZ R0 LOOP
```

How many iterations?

R0 is set to 10
R1 is set to 0

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 10
LOOP2:  LD F3 0 R1
        MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 10
        BNEZ R0 LOOP
```

LOOP2
10

How many iterations?

R0 is set to 10
R1 is set to 0

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 10
LOOP2:  LD F3 0 R1
        MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 10
        BNEZ R0 LOOP
```

LOOP2
10

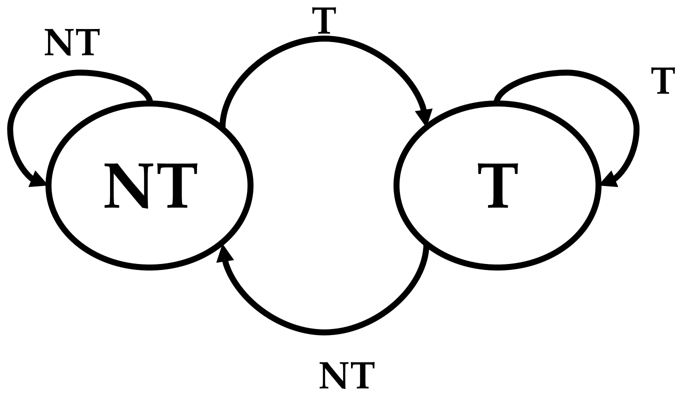
LOOP

No iterations ☺

1bit - BHT - Not Collide

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 10
LOOP2:  LD F3 0 R1
        MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 10
        BNEZ R0 LOOP
```

R0 is set to 10
R1 is set to 0



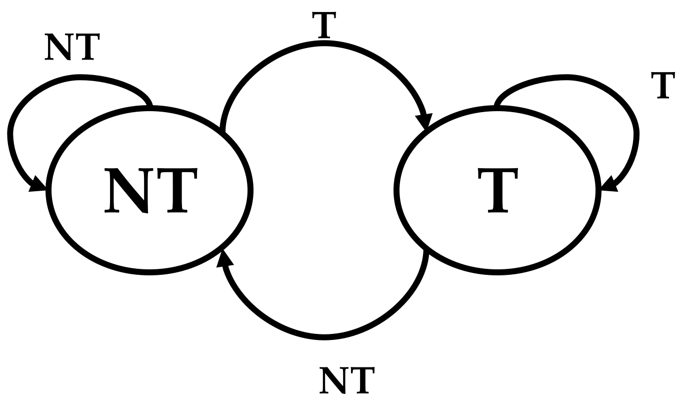
Let us consider that the branch addresses do not collide

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

1bit - BHT - Not Collide

```
LOOP:     LD F3 0 R0
          ADDD F1 F3 F3
          MULTD F2 F3 F1
          ADDI R1 R1 10
LOOP2:    LD F3 0 R1
          MULTD F2 F2 F3
          SUBI R1 R1 1
          BNEZ R1 LOOP2
          SUBI R0 R0 10
          BNEZ R0 LOOP
```

R0 is set to 10
R1 is set to 0



Let us consider that the branch addresses do not collide

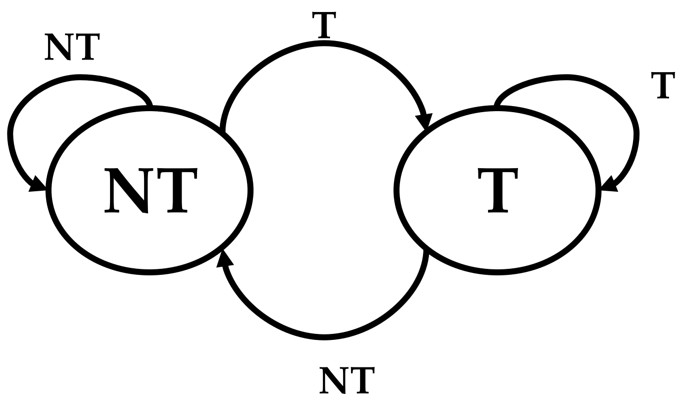
	1-BHT	1-BHT	1-BHT	1-BHT
LOOP:	T	T	NT	NT
LOOP2:	T	NT	T	NT

LOOP → single misprediction
LOOP2 → single misprediction at the end of the loop vs two mispredictions

1bit - BHT - Not Collide

```
LOOP:     LD F3 0 R0
          ADDD F1 F3 F3
          MULTD F2 F3 F1
          ADDI R1 R1 10
LOOP2:    LD F3 0 R1
          MULTD F2 F2 F3
          SUBI R1 R1 1
          BNEZ R1 LOOP2
          SUBI R0 R0 10
          BNEZ R0 LOOP
```

R0 is set to 10
R1 is set to 0


$$\begin{tikzcd} NT \arrow[loop left, "NT"] \arrow[r, bend left, "T"] & T \arrow[loop right, "T"] \arrow[l, bend left, "NT"] \end{tikzcd}$$

Let us consider that the branch addresses do not collide

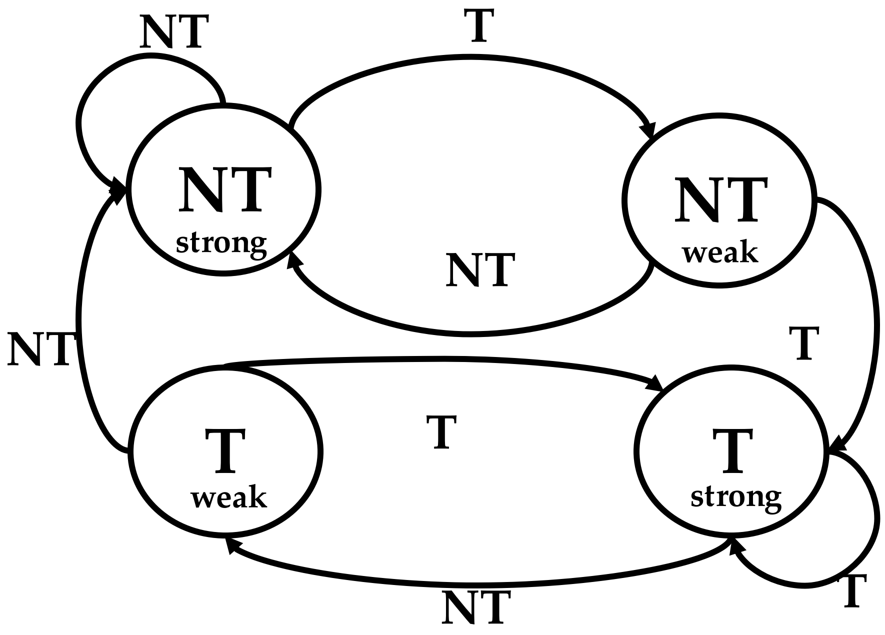
	1-BHT	1-BHT	1-BHT	1-BHT
LOOP:	T	T	NT	NT
LOOP2:	T	NT	T	NT

LOOP → no misprediction
LOOP2 → single misprediction at the end of the loop vs two mispredictions

2bit - BHT

R0 is set to 10
R1 is set to 0

```
LOOP:    LD F3 0 R0
         ADDD F1 F3 F3
         MULTD F2 F3 F1
         ADDI R1 R1 10
LOOP2:   LD F3 0 R1
         MULTD F2 F2 F3
         SUBI R1 R1 1
         BNEZ R1 LOOP2
         SUBI R0 R0 10
         BNEZ R0 LOOP
```



2-BHT

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

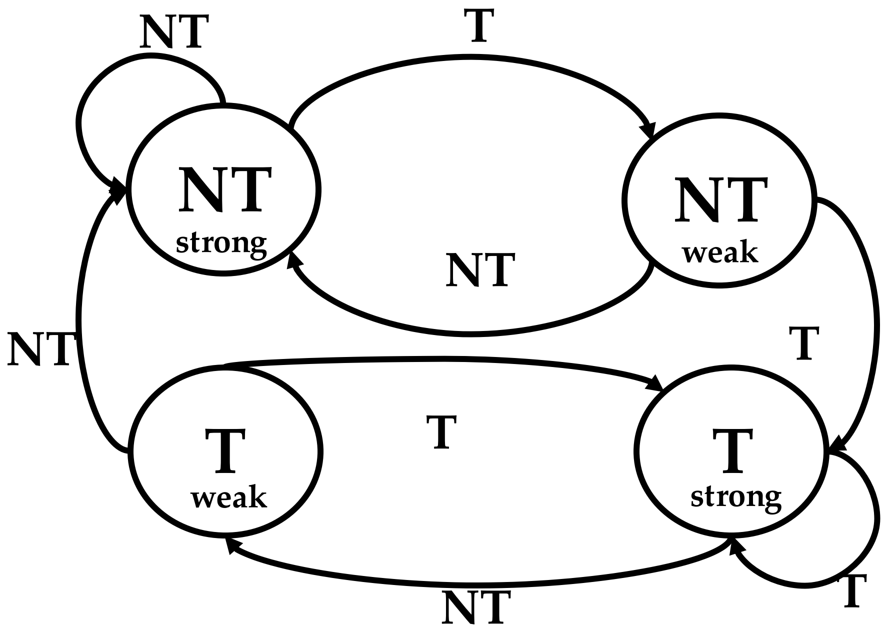
2-BHT

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

2bit - BHT

R0 is set to 10
R1 is set to 0

```
LOOP:   LD F3 0 R0
        ADDD F1 F3 F3
        MULTD F2 F3 F1
        ADDI R1 R1 10
LOOP2:  LD F3 0 R1
        MULTD F2 F2 F3
        SUBI R1 R1 1
        BNEZ R1 LOOP2
        SUBI R0 R0 10
        BNEZ R0 LOOP
```



2-BHT

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

2 + 1 misprediction for LOOP2
1 misprediction for LOOP.

2-BHT

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

1 misprediction for LOOP2
0 misprediction for LOOP.

SUMMARY

Assumption: NO collision

WORST CASES

1-BHT

LOOP:	T
LOOP2:	NT

2 **misprediction for LOOP2**
1 **misprediction for LOOP.**

2-BHT

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

2+1 **misprediction for LOOP2**
1 **misprediction for LOOP.**

BEST CASES

1-BHT

LOOP:	NT
LOOP2:	T

1 **misprediction for LOOP2**
0 **for LOOP**

2-BHT

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

1 **misprediction for LOOP2**
0 **for LOOP**

SUMMARY

Assumption: NO collision

WORST CASES

WORST 2BHT NOT better than BEST 1BHT

1-BHT

LOOP:	T
LOOP2:	NT

2-BHT

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

2 misprediction for LOOP2
1 misprediction for LOOP.

2 1 misprediction for LOOP2
1 misprediction for LOOP.

BEST CASES

(but consider not real nested loop)

1-BHT

LOOP:	NT
LOOP2:	T

2-BHT

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

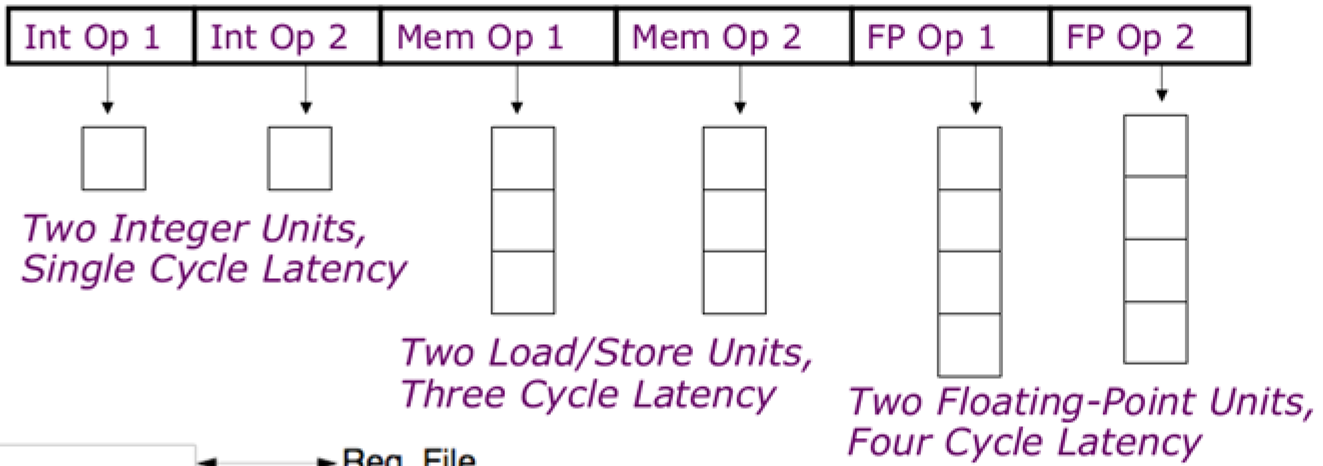
1 misprediction for LOOP2
0 for LOOP

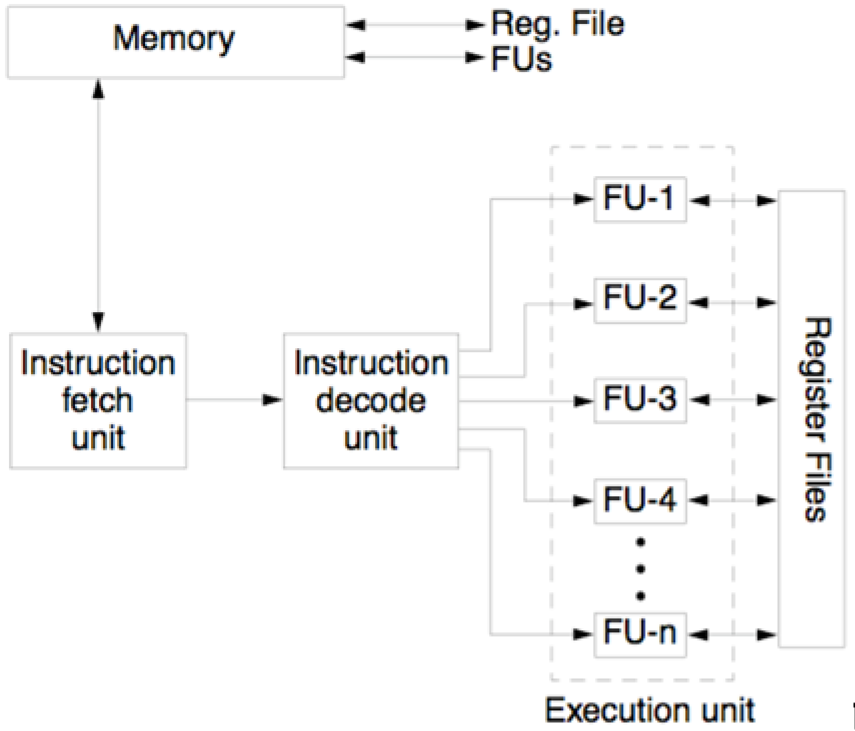
1 misprediction for LOOP2
0 for LOOP

POLITECNICO MILANO 1863 NECST laboratory

POLITECNICO MILANO 1863

Recall VLIW and Static Scheduling


Int Op 1
Int Op 2
Mem Op 1
Mem Op 2
FP Op 1
FP Op 2
Two Integer Units, Single Cycle Latency
Two Load/Store Units, Three Cycle Latency
Two Floating-Point Units, Four Cycle Latency



Static Scheduling: Rely on software for identifying potential parallelism (example of List Based Scheduling with ASAP)

VLIW (Very Long Instruction Word) processors expect dependency-free code.

Exe 1 VLIW: Architecture

- Consider the program be executed on a **3-issue** VLIW MIPS (Very Long Instruction Word) architecture with **3 fully pipelined functional** units
- **Integer ALU** with **1** cycle latency
- **Memory Unit** with **2** cycle latency
- **Floating Point Unit** with **3** cycle latency
- Branch solved in EXE stage, **no early evaluation**

Exe 1 VLIW: Architecture

- Consider the program be executed on a **3-issue** VLIW MIPS (Very Long Instruction Word) architecture with **3 fully pipelined functional** units
- **Integer ALU** with **1** cycle latency
- **Memory Unit** with **2** cycle latency
- **Floating Point Unit** with **3** cycle latency
- Branch solved in EXE stage,

  <u>**no early evaluation**</u>

Exe VLIW.1: schedule

- Considering **one iteration** of the loop
- schedule the assembly code for the 3-issue VLIW machine in the following table by using the **list-based scheduling** with **ASAP**
- Calculate the **performance** (FLOPs per cycle)

- **Do not** use neither software pipelining nor loop unrolling nor modifying loop indexes
- Please do not need to write in NOPs (can leave blank)

Exe VLIW.2: schedule

- **Unroll** the loop by one iteration (so two iterations of the original loop are performed for every branch in the new assembly code)
- You only need to worry about the steady-state code in the core of the loop (no epilogue or prologue)
- schedule the assembly code for the 3-issue VLIW machine in the following table by using the **list-based scheduling** with **ASAP**
- Calculate the **performance** (FLOPs per cycle)

- **Do not** use software pipelining
- Please do not need to write in NOPs (can leave blank)

Exe VLIW.1: schedule

- Considering **one iteration** of the loop
- Schedule the assembly code for the 3-issue VLIW machine in the following table by using the **list-based scheduling** with **ASAP**
- Calculate the **performance** (FLOPs per cycle)

- **Do not** use neither software pipelining nor loop unrolling nor modifying loop indexes
- Please do not need to write in NOPs (can leave blank)

Exe VLIW.1: the code

C Code:

```
for(int i=0; i<N; i++) {
      C[i] = A[i]*A[i] + B[i];
}
```

Assembly Code:

Exe VLIW.1: the code

C Code:

```
for(int i=0; i<N; i++) {
    C[i] = A[i]*A[i] + B[i];
}
```

Assembly Code:

```
loop:  ld     f1, 0(r1)
       ld     f2, 0(r2)
       fmul   f1, f1, f1
       fadd   f1, f1, f2
       st     f1, 0(r3)
       addi   r1, r1, 4
       addi   r2, r2, 4
       addi   r3, r3, 4
       bne    r3, r4, loop
```

Exe VLIW.1: schedule

```
loop:  ld     f1, 0(r1)
       ld     f2, 0(r2)
       fmul   f1, f1, f1
       fadd   f1, f1, f2
       st     f1, 0(r3)
       addi   r1, r1, 4
       addi   r2, r2, 4
       addi   r3, r3, 4
       bne    r3, r4, loop
```

ALU	1 cc	
MU		2 cc
FPU	3 cc	

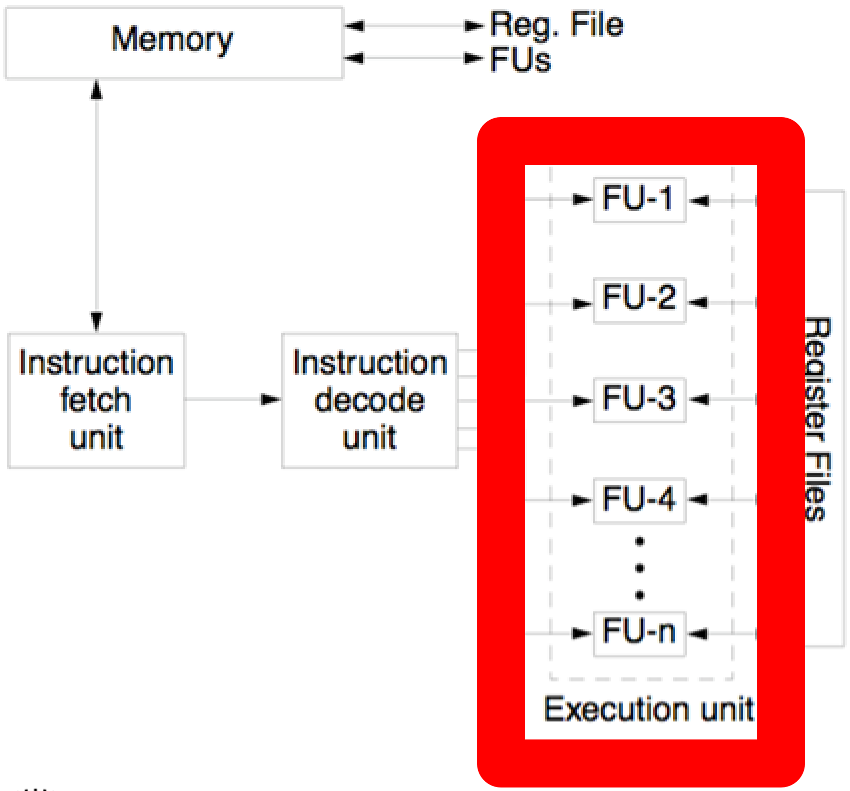
POLITECNICO MILANO 1863 NECST laboratory

POLITECNICO MILANO 1863

Exe VLIW.1: schedule

```
loop:   ld      f1, 0(r1)
        ld      f2, 0(r2)
        fmul    f1, f1, f1
        fadd    f1, f1, f2
        st      f1, 0(r3)
        addi    r1, r1, 4
        addi    r2, r2, 4
        addi    r3, r3, 4
        bne     r3, r4, loop
```

ALU	1 cc	
MU		2 cc
FPU	3 cc	



Exe VLIW.1: schedule

```
: ld f1, 0(r1)
  ld f2, 0(r2)
  fmul f1, f1, f1
  fadd f1, f1, f2
  st f1, 0(r3)
  addi r1, r1, 4
  addi r2, r2, 4
  addi r3, r3, 4
  bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1			
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
    : ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f2, 0(r2)	
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
  ➡   fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1) 1cc	
C2		ld f2, 0(r2) 2cc	
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1) 1cc	
C2		ld f2, 0(r2) 2cc	
C3			fmul f1, f1, f1
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f2, 0(r2)	
C3			fmul f1, f1, f1 1cc
C4			2cc
C5			3cc
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

POLITECNICO MILANO 1863 NECST laboratory

POLITECNICO MILANO 1863

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f2, 0(r2)	
C3			fmul f1, f1, f1 1cc
C4			2cc
C5			3cc
C6			fadd f1, f1, f2
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2 1cc
C7			2cc
C8			3cc
C9		st f1, 0(r3)	
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
  ➡   addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1	addi r1, r1, 4	ld f1, 0(r1)	
C2		ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2
C7			
C8			
C9		st f1, 0(r3)	
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1	addi r1, r1, 4	ld f1, 0(r1)	
C2	addi r2, r2, 4	ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2
C7			
C8			
C9		st f1, 0(r3)	
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1	addi r1, r1, 4	ld f1, 0(r1)	
C2	addi r2, r2, 4	ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2
C7			
C8			
C9	addi r3, r3, 4	st f1, 0(r3)	
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
→     bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1	addi r1, r1, 4	ld f1, 0(r1)	
C2	addi r2, r2, 4	ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2
C7			
C8			
C9	addi r3, r3, 4	st f1, 0(r3)	
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: schedule

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1	addi r1, r1, 4	ld f1, 0(r1)	
C2	addi r2, r2, 4	ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2
C7			
C8			
C9	addi r3, r3, 4	st f1, 0(r3)	
C10	bne r3, r4, loop		
C11			
C12			
C13			
C14			
C15			

Exe VLIW.1: FLOPs/CC

```
loop: ld f1, 0(r1)
      ld f2, 0(r2)
      fmul f1, f1, f1
      fadd f1, f1, f2
      st f1, 0(r3)
      addi r1, r1, 4
      addi r2, r2, 4
      addi r3, r3, 4
      bne r3, r4, loop
```

FLOPs/CC=

= 2 / 10

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1	addi r1, r1, 4	ld f1, 0(r1)	
C2	addi r2, r2, 4	ld f2, 0(r2)	
C3			fmul f1, f1, f1
C4			
C5			
C6			fadd f1, f1, f2
C7			
C8			
C9	addi r3, r3, 4	st f1, 0(r3)	
C10	bne r3, r4, loop		
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

- **Unroll** the loop by one iteration (so two iterations of the original loop are performed for every branch in the new assembly code)
- You only need to worry about the steady-state code in the core of the loop (no epilogue or prologue)
- schedule the assembly code for the 3-issue VLIW machine in the following table by using the **list-based scheduling** with **ASAP**
- Calculate the **performance** (FLOPs per cycle)

- **Do not** use software pipelining
- Please do not need to write in NOPs (can leave blank)

Exe VLIW.2: the code

C Code:

```
for(int i=0; i<N; i+=2) {
    C[i] = A[i]*A[i] + B[i];
    C[i+1] = A[i+1]*A[i+1] + B[i+1];
}
```

Assembly Code:

Exe VLIW.2: the code

C Code:

```
for(int i=0; i<N; i+=2) {
    C[i] = A[i]*A[i] + B[i];
    C[i+1] = A[i+1]*A[i+1] + B[i+1];
}
```

Assembly Code:

```
loop:   ld      f1, 0(r1)
        ld      f3, 4(r1)
        ld      f2, 0(r2)
        ld      f4, 4(r2)
        fmul    f1, f1, f1
        fmul    f3, f3, f3
        fadd    f1, f1, f2
        fadd    f3, f3, f4
        st      f1, 0(r3)
        st      f3, 4(r3)
        addi    r1, r1, 8
        addi    r2, r2, 8
        addi    r3, r3, 8
        bne     r3, r4, loop
```

Exe VLIW.2: schedule

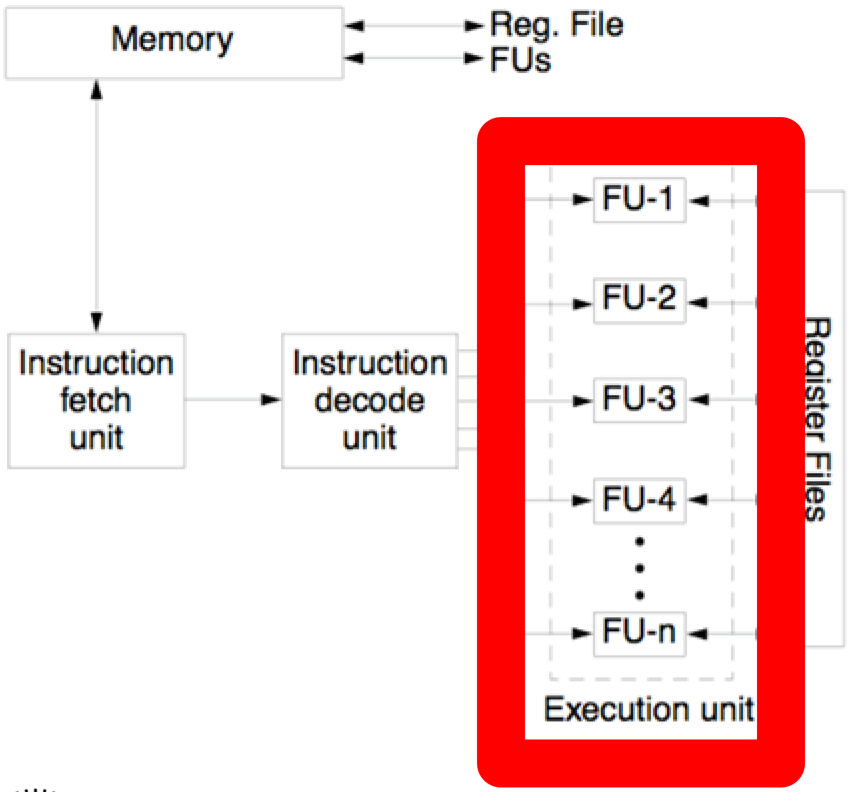
```
loop:   ld      f1, 0(r1)
        ld      f3, 4(r1)
        ld      f2, 0(r2)
        ld      f4, 4(r2)
        fmul    f1, f1, f1
        fmul    f3, f3, f3
        fadd    f1, f1, f2
        fadd    f3, f3, f4
        st      f1, 0(r3)
        st      f3, 4(r3)
        addi    r1, r1, 8
        addi    r2, r2, 8
        addi    r3, r3, 8
        bne     r3, r4, loop
```

ALU	1 cc	
MU		2 cc
FPU	3 cc	

Exe VLIW.2: schedule

```
loop:  ld     f1, 0(r1)
       ld     f3, 4(r1)
       ld     f2, 0(r2)
       ld     f4, 4(r2)
       fmul   f1, f1, f1
       fmul   f3, f3, f3
       fadd   f1, f1, f2
       fadd   f3, f3, f4
       st     f1, 0(r3)
       st     f3, 4(r3)
       addi   r1, r1, 8
       addi   r2, r2, 8
       addi   r3, r3, 8
       bne    r3, r4, loop
```

ALU	1 cc
MU	2 cc
FPU	3 cc



Exe VLIW.2: schedule

```
: ld   f1, (r1)
  ld   f3, 4(r1)
  ld   f2, 0(r2)
  ld   f4, 4(r2)
  fmul f1, f1, f1
  fmul f3, f3, f3
  fadd f1, f1, f2
  fadd f3, f3, f4
  st   f1, 0(r3)
  st   f3, 4(r3)
  addi r1, r1, 8
  addi r2, r2, 8
  addi r3, r3, 8
  bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1			
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
ld   f1, (r1)
ld   f3, 4(r1)
ld   f2, 0(r2)
ld   f4, 4(r2)
fmul f1, f1, f1
fmul f3, f3, f3
fadd f1, f1, f2
fadd f3, f3, f4
st   f1, 0(r3)
st   f3, 4(r3)
addi r1, r1, 8
addi r2, r2, 8
addi r3, r3, 8
bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3		ld f2, 0(r2)	
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3		ld f2, 0(r2)	
C4		ld f4, 4(r2)	
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1) 1cc	
C2		ld f3, 4(r1) 2cc	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4		ld f4, 4(r2)	
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld    f1, (r1)
      ld    f3, 4(r1)
      ld    f2, 0(r2)
      ld    f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st    f1, 0(r3)
      st    f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1) 1cc	
C3		ld f2, 0(r2) 2cc	fmul f1, f1, f1
C4		ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
  ➡   fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1 1cc
C4		ld f4, 4(r2)	fmul f3, f3, f3 2cc
C5			3cc
C6			fadd f1, f1, f2
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
  ➡   fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4		ld f4, 4(r2)	1cc fmul f3, f3, f3
C5			2cc
C6			fadd f1, f1, f2 3cc
C7			fadd f3, f3, f4
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4		ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2 1cc
C7			fadd f3, f3, f4 2cc
C8			3cc
C9		st f1, 0(r3)	
C10			
C11			
C12			
C13			
C14			
C15			

POLITECNICO MILANO 1863

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2		ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4		ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2
C7			fadd f3, f3, f4 1cc
C8			2cc
C9		st f1, 0(r3)	3cc
C10		st f3, 4(r3)	
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2	addi r1, r1, 8	ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4		ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2
C7			fadd f3, f3, f4
C8			
C9		st f1, 0(r3)	
C10		st f3, 4(r3)	
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2	addi r1, r1, 8	ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4	addi r2, r2, 8	ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2
C7			fadd f3, f3, f4
C8			
C9		st f1, 0(r3)	
C10		st f3, 4(r3)	
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2	addi r1, r1, 8	ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4	addi r2, r2, 8	ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2
C7			fadd f3, f3, f4
C8			
C9		st f1, 0(r3)	
C10	addi r3, r3, 8	st f3, 4(r3)	
C11			
C12			
C13			
C14			
C15			

Exe VLIW.2: schedule

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2	addi r1, r1, 8	ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4	addi r2, r2, 8	ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2
C7			fadd f3, f3, f4
C8			
C9		st f1, 0(r3)	
C10	addi r3, r3, 8	st f3, 4(r3)	
C11	bne r3, r4, loop		
C12			
C13			
C14			
C15			

POLITECNICO MILANO 1863

Exe VLIW.2: FLOPs/CC

```
loop: ld   f1, (r1)
      ld   f3, 4(r1)
      ld   f2, 0(r2)
      ld   f4, 4(r2)
      fmul f1, f1, f1
      fmul f3, f3, f3
      fadd f1, f1, f2
      fadd f3, f3, f4
      st   f1, 0(r3)
      st   f3, 4(r3)
      addi r1, r1, 8
      addi r2, r2, 8
      addi r3, r3, 8
      bne  r3, r4, loop
```

FLOPs/CC=

= 4 / 11

	Integer ALU (1 cc)	Memory Unit (2 cc)	FPU (3 cc)
C1		ld f1, 0(r1)	
C2	addi r1, r1, 8	ld f3, 4(r1)	
C3		ld f2, 0(r2)	fmul f1, f1, f1
C4	addi r2, r2, 8	ld f4, 4(r2)	fmul f3, f3, f3
C5			
C6			fadd f1, f1, f2
C7			fadd f3, f3, f4
C8			
C9		st f1, 0(r3)	
C10	addi r3, r3, 8	st f3, 4(r3)	
C11	bne r3, r4, loop		
C12			
C13			
C14			
C15			

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

and are ***properties of their respective owners***