POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

POLITECNICO MILANO 1863 NECST laboratory

Exercise Session 5

Dynamic Scheduling Scoreboard, Dynamic Scheduling Tomasulo, Complex Pipeline, (Extra:Simple Scheduling)

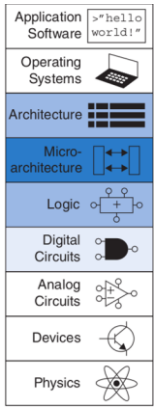
Advanced Computer Architectures

7th April 2025

Davide Conficconi ‹davide.conficconi@polimi.it›

POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

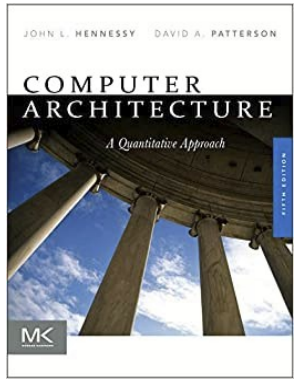
Recall: Material (EVERYTHING OPTIONAL)



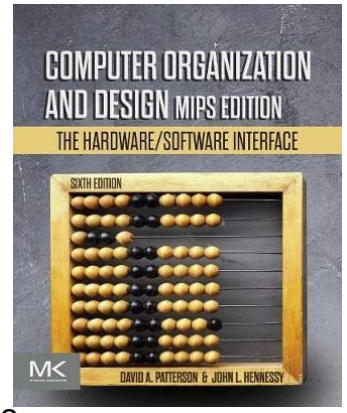
https://webeep.polimi.it/course/view.php?id=14754

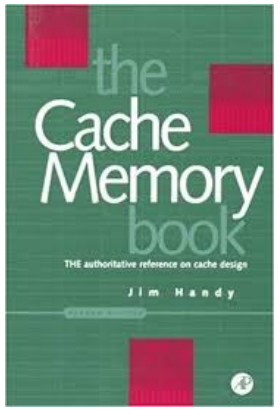
https://tinyurl.com/aca-grid25

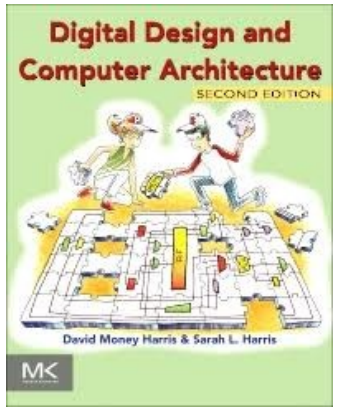
Textbook: Hennessy and Patterson, Computer Architecture: A Quantitative Approach

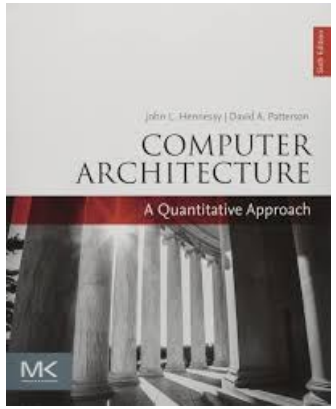


Other Interesting Reference







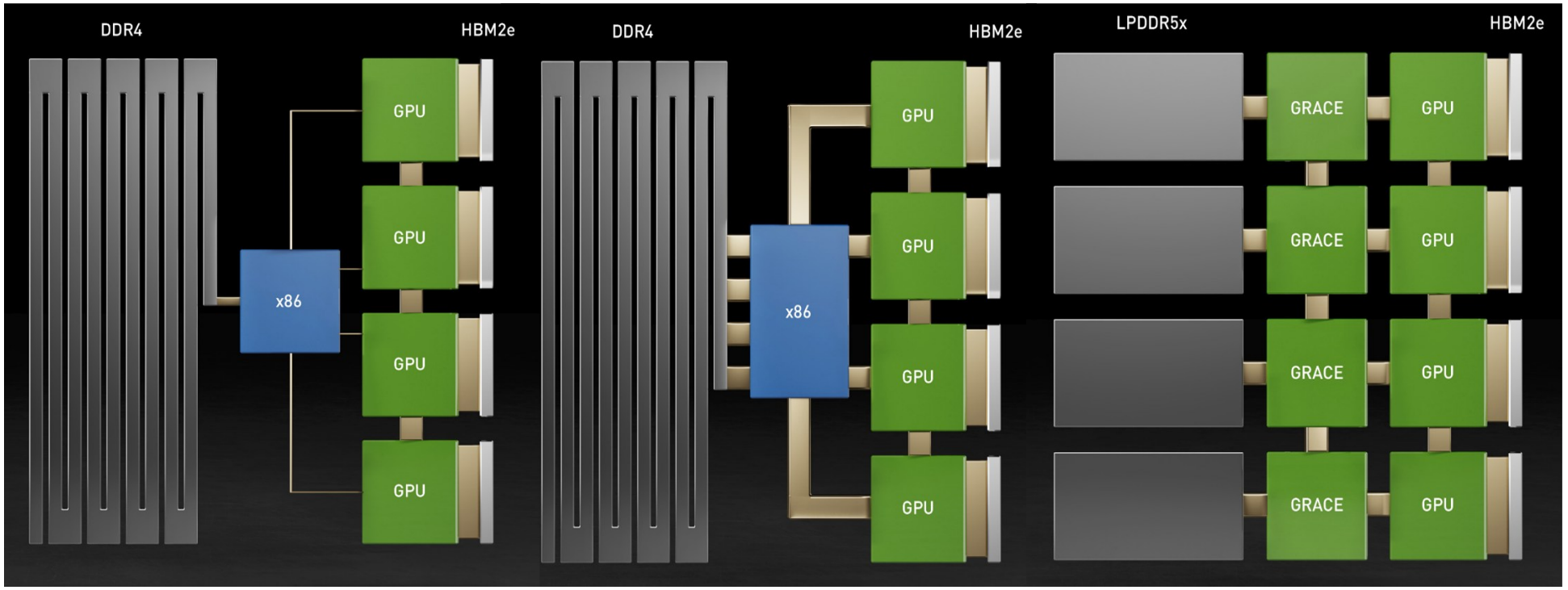


Recall 2021: NVIDIA Grace

https://www.youtube.com/watch?v=eAn[redacted]oiZwUXA and
https://www.nvidia.com/en-us/gtc/

New ARM-based CPU Grace (named after Grace Hopper, she was CS and pioneered computer programming)

Deliver high system-to-memory bandwidth (performance)



Sources: GTC Keynote Slides

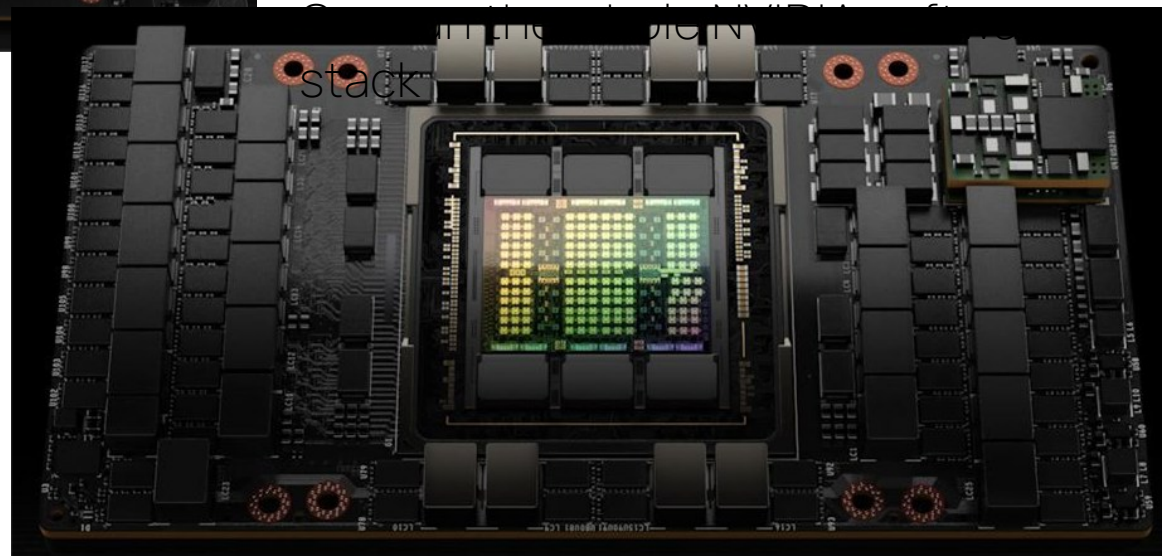
Additional info may be found at: https://www.nextplatform.com/2021/04/12/nvidia-enters-the-arms-race-with-homegrown-grace-cpus/?mc[redacted]cid=aa58a09ac6&mc[redacted]eid=2acaf941b8

Recall 2022: News from the outer world: Grace(CPU) & Hopper (GPU)

https://nvidianews.nvidia.com/press-kit/gtc-2022

144 High-Performance Cores and 1 Terabyte/Second Memory; Doubles Performance and Energy-Efficiency of Server Chips (source)

2x mem bw
2 CPU via NVLink Cache Coherent
stack



The New Engine for World's AI Infrastructure, NVIDIA H100 GPU Makes Order of Magnitude Performance Leap (source)

PCIe Gen 5 and HBM3 support
Specialized Transformer Engines

Recall 2023: News from the outer world:
Grace CPU Available (Jan 23)

https://developer.nvidia.com/blog/nvidia-grace-cpu-superchip-architecture-in-

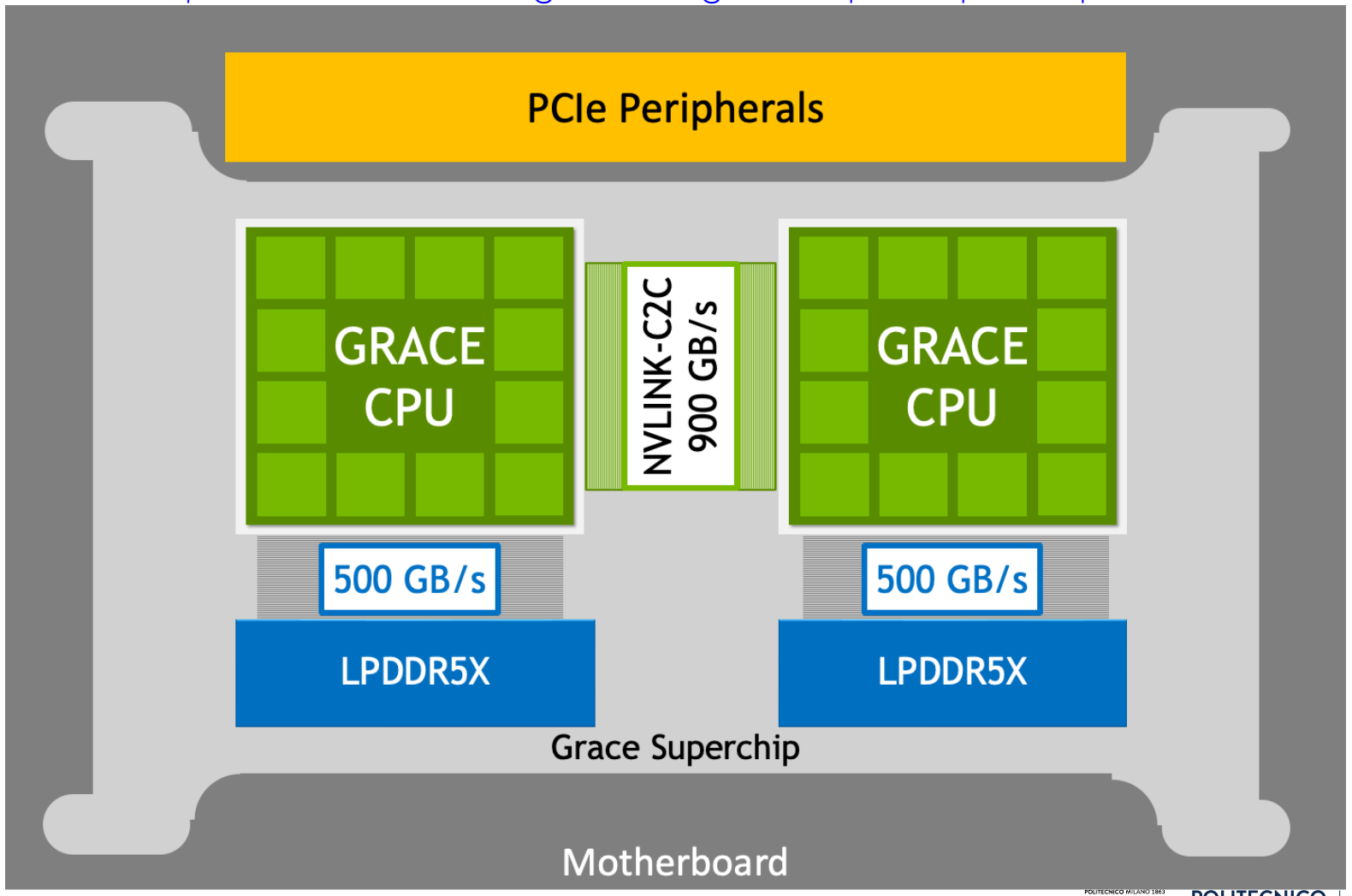


Image from: NVIDIA

Recall 2023: News from the outer world:
Grace CPU Results at GTC'23 Spring

https://blogs.nvidia.com/blog/2023/03/21/grace-cpu-energy-efficiency/

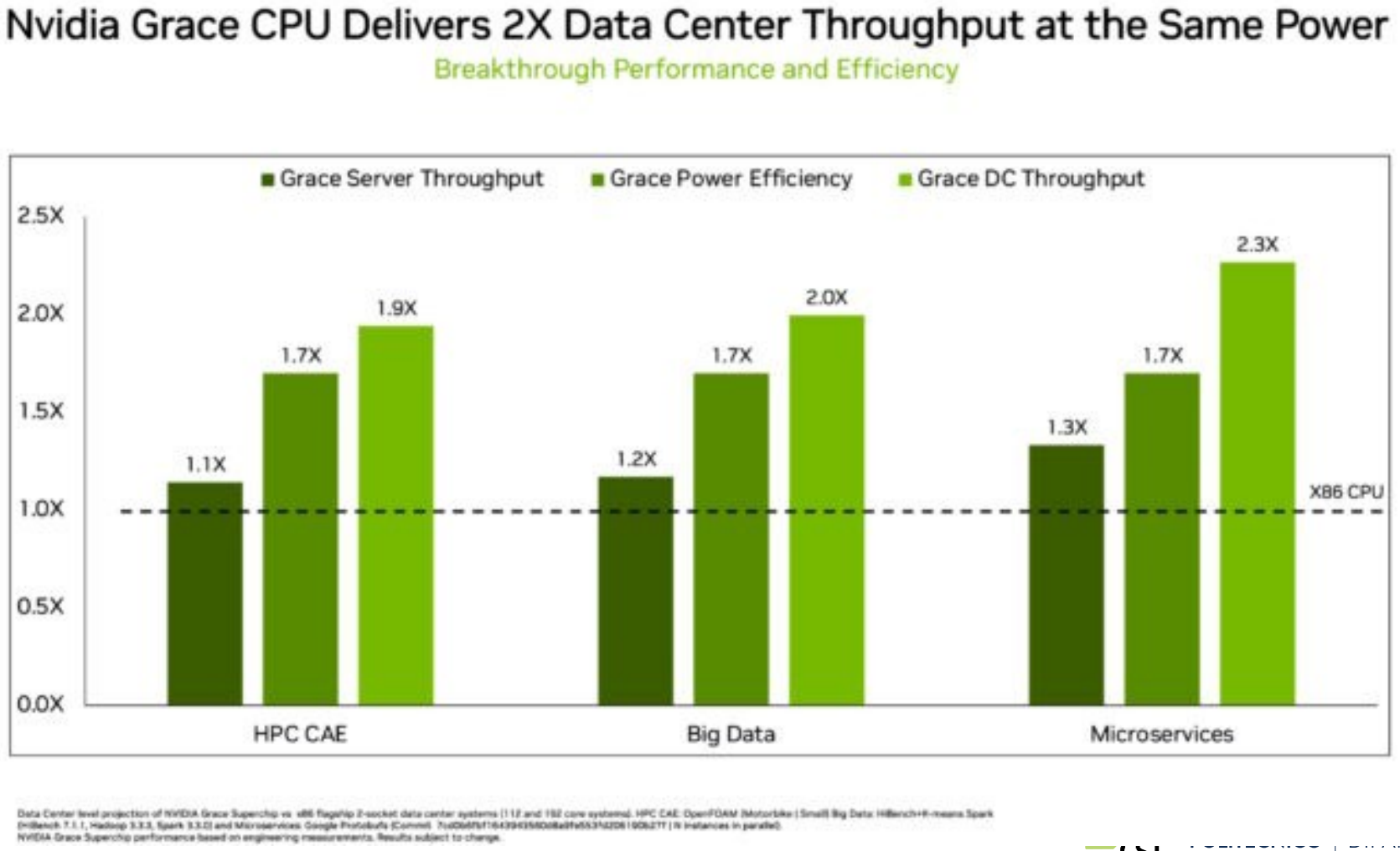
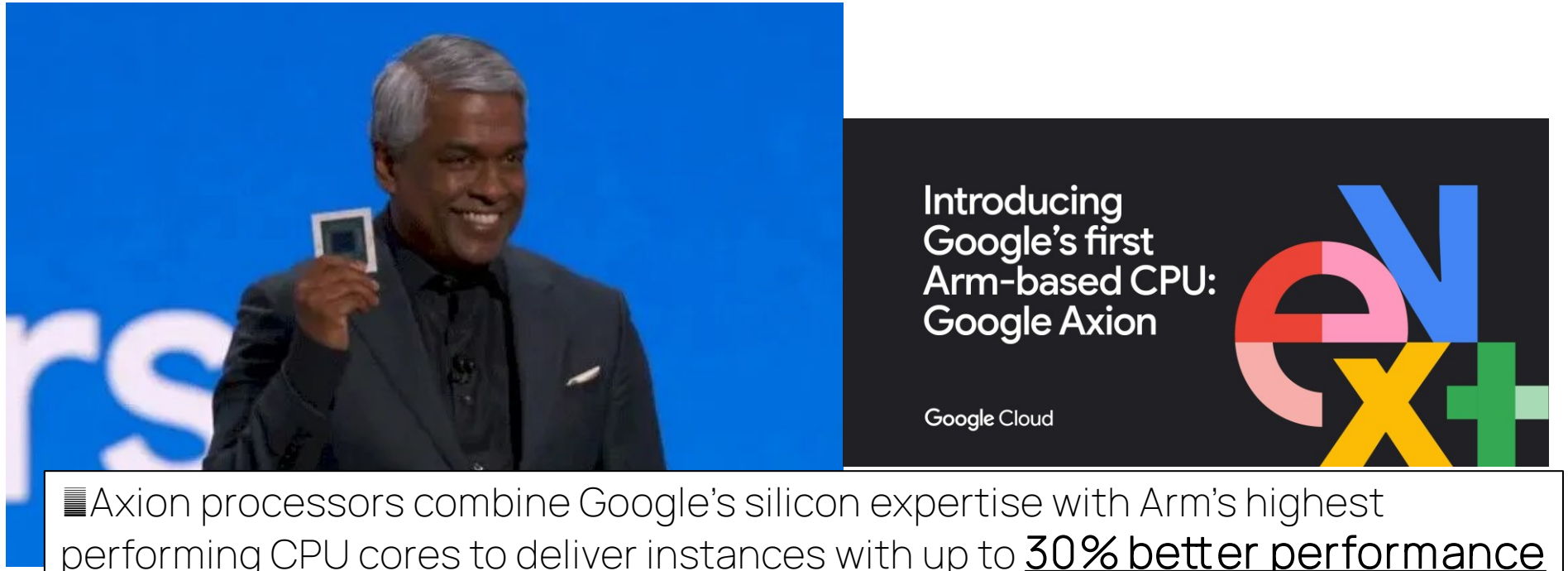


Image from: NVIDIA

Recall 2024: News from the outer world:
Google Axion Processors (ARM-based)

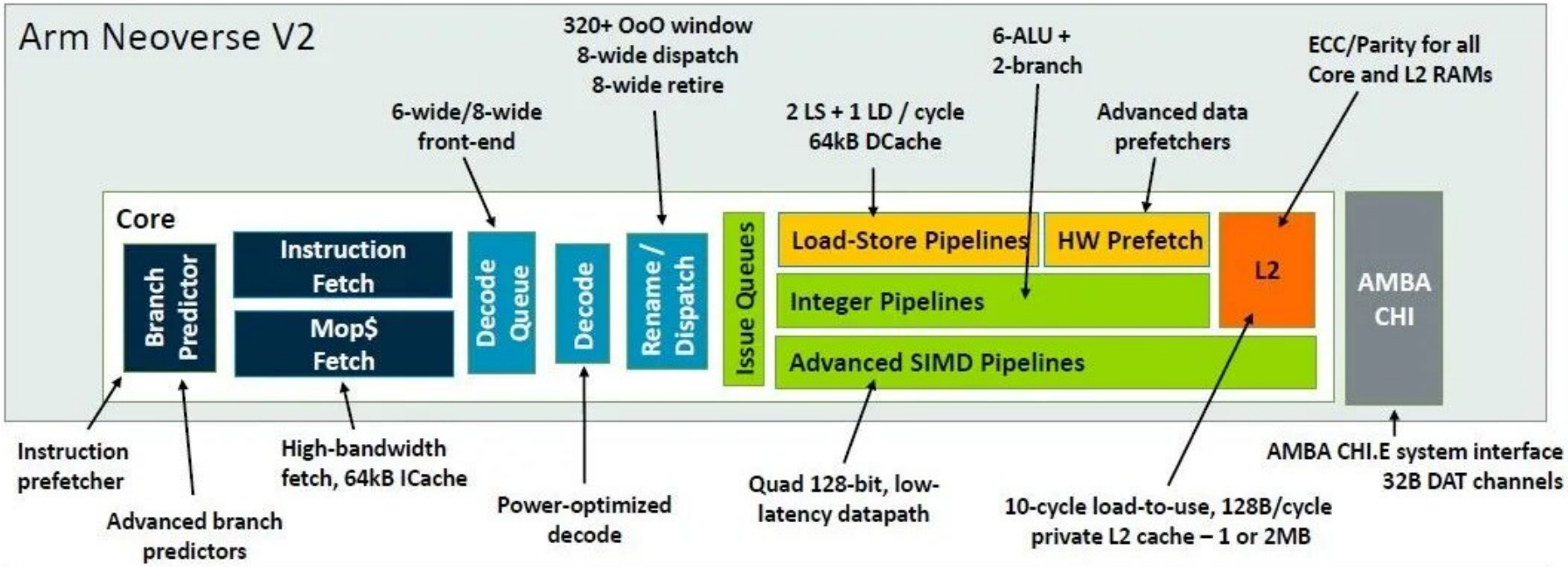
https://cloud.google.com/blog/products/compute/introducing-googles-new-arm-based-cpu
https://www.nextplatform.com/2024/04/09/google-joins-the-homegrown-arm-server-cpu-club/



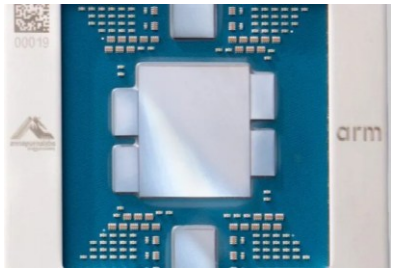
▓Axion processors combine Google's silicon expertise with Arm's highest performing CPU cores to deliver instances with up to 30% better performance than the fastest general-purpose Arm-based instances available in the cloud today, up to 50% better performance and up to 60% better energy efficiency than comparable current-generation X86-based instances.▓

Recall 2024: News from the outer world: Other ARM-based Datacenter CPUs

https://www.nextplatform.com/2024/04/09/google-joins-the-homegrown-arm-server-cpu-club/



https://www.nextplatform.com/2023/11/28/aws-adopts-arm-v2-cores-for-expansive-graviton4-server-cpu/



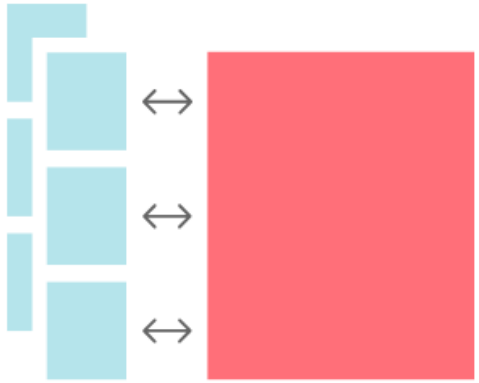
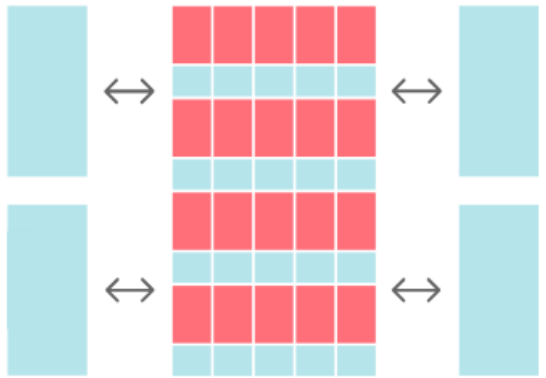
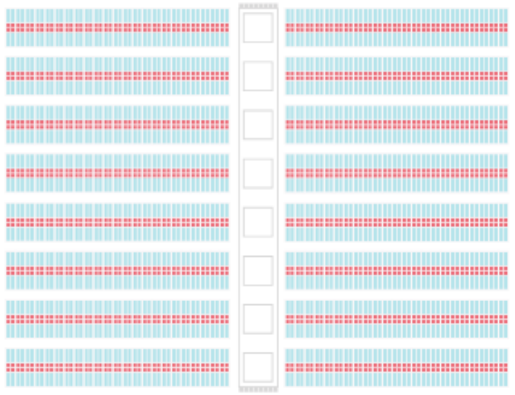
News from the outer world: Vera CPU – Rubin GPU Announced

https://www.nextplatform.com/2025/03/19/nvidia-draws-gpu-system-roadmap-out-to-2028/

News from the outer world: SoftBank Just Bought Ampere Computing

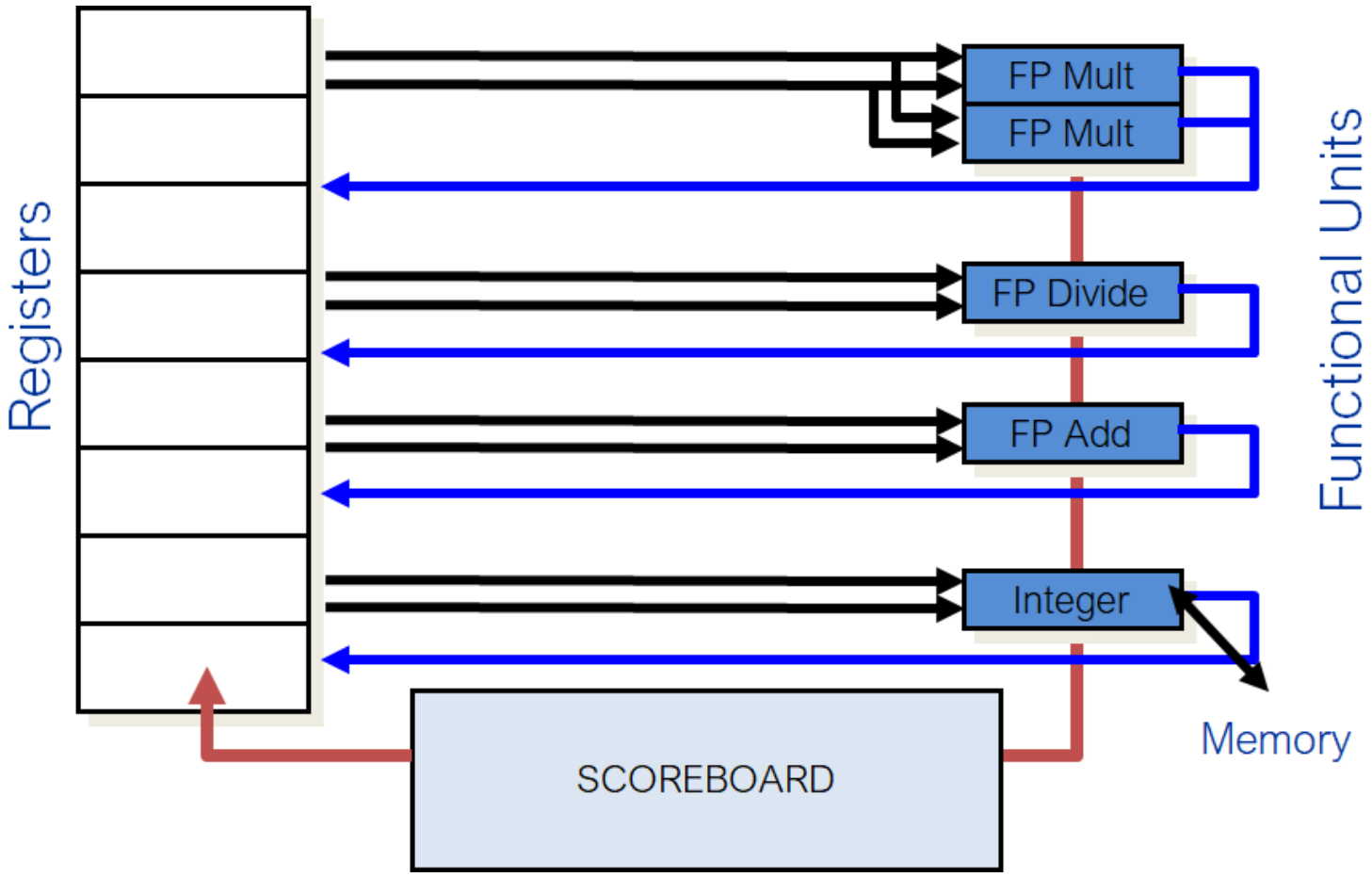
https://www.nextplatform.com/2025/03/21/why-did-softbank-just-buy-ampere-computing/

News from the outer world: SoftBank Bought ~~Ampere Computing~~ Graphcore

	CPU	GPU	IPU
Parallelism	Designed for scalar processes	SIMD/SIMT architecture. Designed for large blocks of dense contiguous data	Massively parallel MIMD. Designed for fine-grained, high-performance computing
			
Memory Access	Off-chip memory	Model and data spread across off-chip and small on-chip cache, and shared memory	Model and data tightly coupled, and large locally distributed SRAM

https://www.nextplatform.com/2024/07/12/can-softbank-be-an-ai-supercomputer-player-will-arm-lend-a-hand/

Recall: Exe Scoreboard

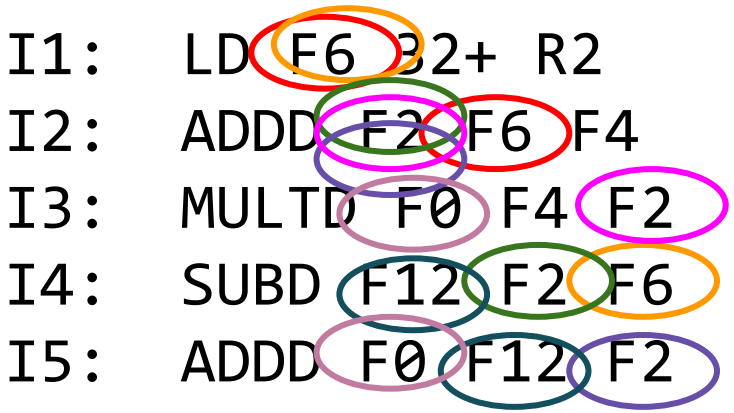


Parallel operation in the control data 6600

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; WAR if need to read	Notify Scoreboard on completion;	WAR &Struct check (FUs will hold results); Can overlap issue/read&write 4 Structural Hazard;

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: exists 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 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 F2	FPU2
I4	SUBD F12 F2 F6	4	11	14		RAW F2	FPU3
I5	ADDD F0 F12 F2					WAW F0	

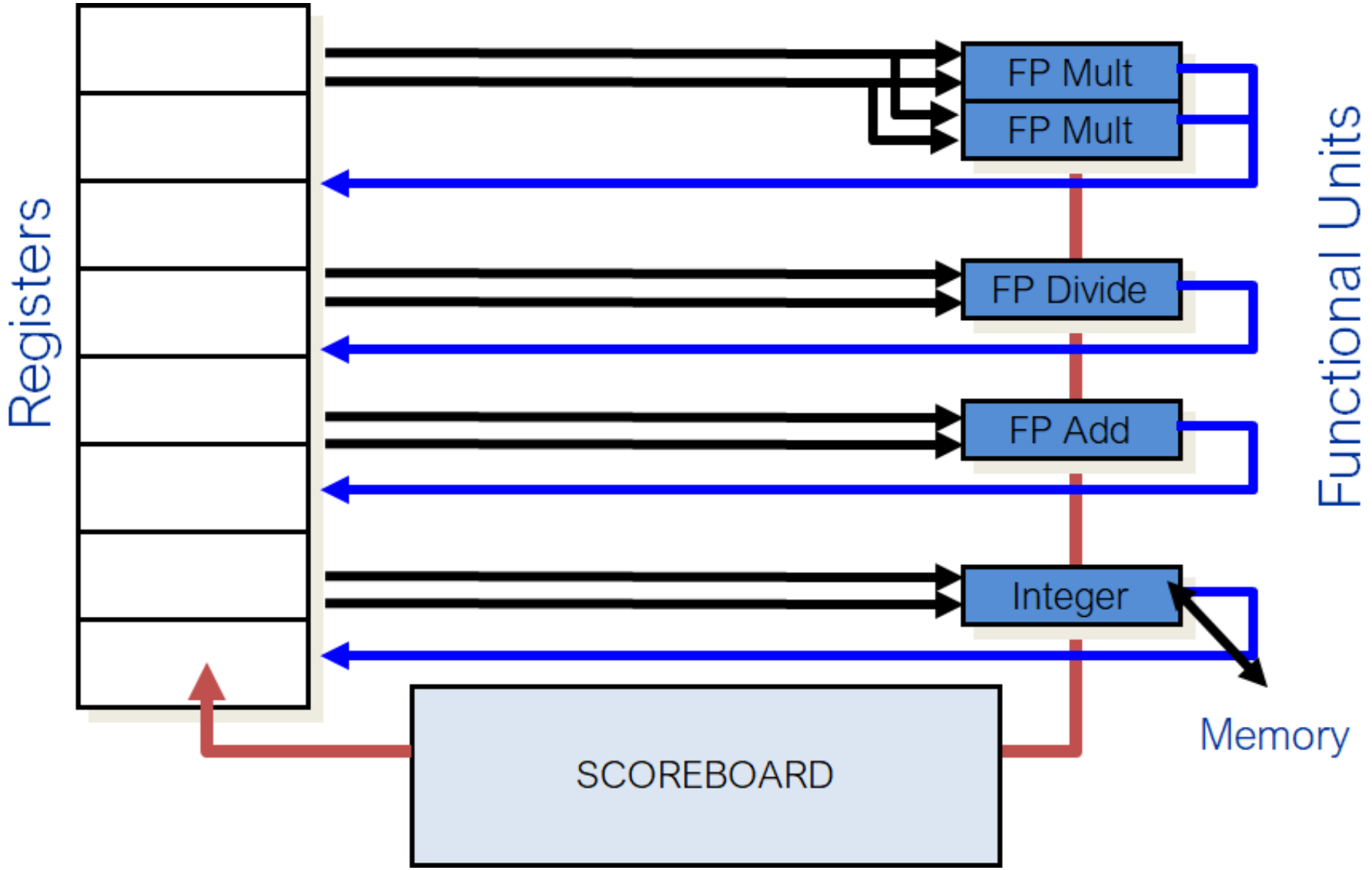
SINGLE WRITE PORT?!?!?!?!?!?!?

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, 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 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, 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 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~~

Recall: Tomasulo Algorithm

Another dynamic algorithm: allows execution to proceed in the presence of dependences

Invented at IBM 3 years after CDC 6600 for the IBM 360/91

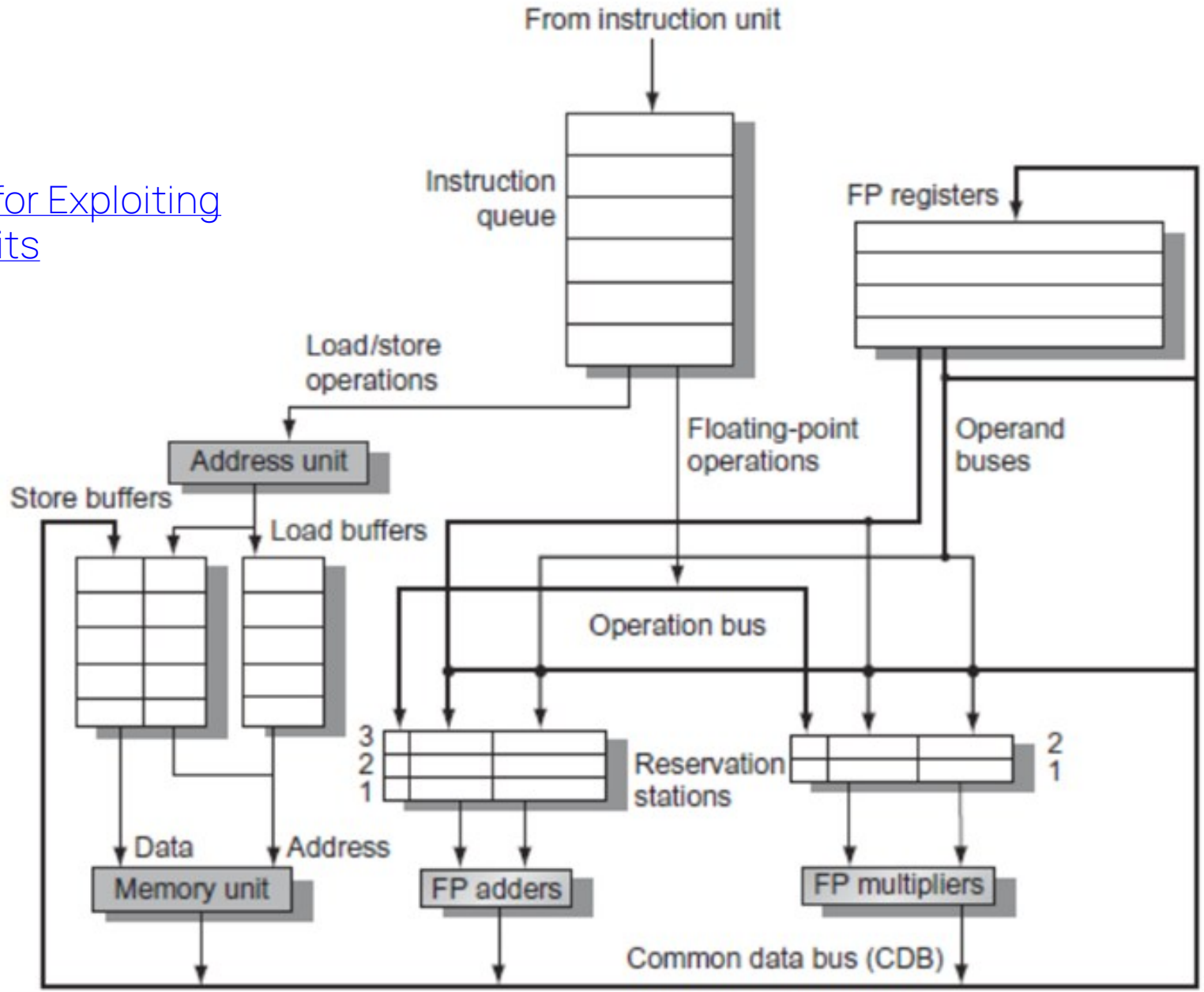
Same Goal: high performance w/o special compilers

Lead to:

Alpha 21264, HP 8000, MIPS 10000, Pentium II, PowerPC 604

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 Tomasulo: 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 Tomasulo: 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 Tomasulo: 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

RAW F2 I2-I5

RAW F12 I4-I5

WAW F0 I3-I5

Exe.2 Tomasulo

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	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`						

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.2 Tomasulo CC 0

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	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`						

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.2 Tomasulo CC 1

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
`I1:LD F6 32+ R2`	1				RS1	
`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
RAW F2 I2-I5
RAW F12 I4-I5
~~**WAW F0 I3-I5**~~

Exe.2 Tomasulo CC 2

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
`I1:LD F6 32+ R2`	1	2			RS1	LDU1
`I2:ADDD F2 F6 F4`	2				RS3	
`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
RAW F2 I2-I5
RAW F12 I4-I5
~~**WAW F0 I3-I5**~~

Exe.2 Tomasulo CC 3

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
`I1:LD F6 32+ R2`	1	2			RS1	LDU1
`I2:ADDD F2 F6 F4`	2			RAW $F6	RS3	
`I3:MULTD F0 F4 F2`	3				RS4	
`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
RAW F2 I2-I5
RAW F12 I4-I5
~~**WAW F0 I3-I5**~~

Exe.2 Tomasulo CC 4

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
`I1:LD F6 32+ R2`	1	2	4		RS1	LDU1
`I2:ADDD F2 F6 F4`	2			RAW $F6	RS3	
`I3:MULTD F0 F4 F2`	3			RAW $F2	RS4	
`I4:SUBD F12 F2 F6`	4				RS5	
`I5:ADDD F0 F12 F2`						

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.2 Tomasulo CC 5

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5		RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3			RAW $F2	RS4	
I4:SUBD F12 F2 F6	4			RAW $F2	RS5	
I5:ADDD F0 F12 F2				Struct RS3	RS3	

~~RAW F6 I1-I2~~ RAW F2 I2-I5

~~RAW F6 I1-I4~~ RAW F12 I4-I5

RAW F2 I2-I3 ~~WAW F0 I3-I5~~

RAW F2 I2-I4

Exe.2 Tomasulo CC 8

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3			RAW $F2	RS4	
I4:SUBD F12 F2 F6	4			RAW $F2	RS5	
I5:ADDD F0 F12 F2				Struct RS3	RS3	

~~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.2 Tomasulo CC 9

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3	9		RAW $F2	RS4	ALU2
I4:SUBD F12 F2 F6	4	9		RAW $F2	RS5	ALU3
I5:ADDD F0 F12 F2	9			Struct RS3	RS3	

~~RAW F6 I1-I2~~ ~~RAW F2 I2-I5~~

~~RAW F6 I1-I4~~ RAW F12 I4-I5

~~RAW F2 I2-I3~~ ~~WAW F0 I3-I5~~

~~RAW F2 I2-I4~~

Exe.2 Tomasulo CC 10

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3	9		RAW $F2	RS4	ALU2
I4:SUBD F12 F2 F6	4	9		RAW $F2	RS5	ALU3
I5:ADDD F0 F12 F2	9			Struct RS3 + RAW $F12	RS3	

~~RAW F6 I1-I2~~ ~~RAW F2 I2-I5~~

~~RAW F6 I1-I4~~ RAW F12 I4-I5

~~RAW F2 I2-I3~~ ~~WAW F0 I3-I5~~

~~RAW F2 I2-I4~~

Exe.2 Tomasulo CC 12

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3	9	12	RAW $F2	RS4	ALU2
I4:SUBD F12 F2 F6	4	9		RAW $F2 + Struct CDB	RS5	ALU3
I5:ADDD F0 F12 F2	9			Struct RS3 + RAW $F12	RS3	

~~RAW F6 I1-I2~~ ~~RAW F2 I2-I5~~

~~RAW F6 I1-I4~~ RAW F12 I4-I5

~~RAW F2 I2-I3~~ ~~WAW F0 I3-I5~~

~~RAW F2 I2-I4~~

Exe.2 Tomasulo CC 13

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3	9	12	RAW $F2	RS4	ALU2
I4:SUBD F12 F2 F6	4	9	13	RAW $F2 + Struct CDB	RS5	ALU3
I5:ADDD F0 F12 F2	9			Struct RS3 + RAW $F12	RS3	

~~RAW F6 I1-I2~~ ~~RAW F2 I2-I5~~

~~RAW F6 I1-I4~~ ~~RAW F12 I4-I5~~

~~RAW F2 I2-I3~~ ~~WAW F0 I3-I5~~

~~RAW F2 I2-I4~~

Exe.2 Tomasulo CC 14

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3	9	12	RAW $F2	RS4	ALU2
I4:SUBD F12 F2 F6	4	9	13	RAW $F2 + Struct CDB	RS5	ALU3
I5:ADDD F0 F12 F2	9	14		Struct RS3 + RAW $F12	RS3	ALU1

~~RAW F6 I1-I2~~ ~~RAW F2 I2-I5~~

~~RAW F6 I1-I4~~ ~~RAW F12 I4-I5~~

~~RAW F2 I2-I3~~ ~~WAW F0 I3-I5~~

~~RAW F2 I2-I4~~

Exe.2 Tomasulo CC 17

- 2 RESERVATION STATIONS (RS1, RS2) + 1 LOAD/STORE unit (LDU1) with latency 2
- 3 RESERVATION STATIONS (RS3, RS4, RS5) + 3 ALU/BR FUs (ALU1, ALU2, ALU3) with latency 3

Instruction	ISSUE	START EXE	WB	Hazards Type	RSi	Unit
I1:LD F6 32+ R2	1	2	4		RS1	LDU1
I2:ADDD F2 F6 F4	2	5	8	RAW $F6	RS3	ALU1
I3:MULTD F0 F4 F2	3	9	12	RAW $F2	RS4	ALU2
I4:SUBD F12 F2 F6	4	9	13	RAW $F2 + Struct CDB	RS5	ALU3
I5:ADDD F0 F12 F2	9	14	17	Struct RS3 + RAW $F12	RS3	ALU1

~~RAW F6 I1-I2~~ ~~RAW F2 I2-I5~~

~~RAW F6 I1-I4~~ ~~RAW F12 I4-I5~~

~~RAW F2 I2-I3~~ ~~WAW F0 I3-I5~~

~~RAW F2 I2-I4~~

Recall Exe.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

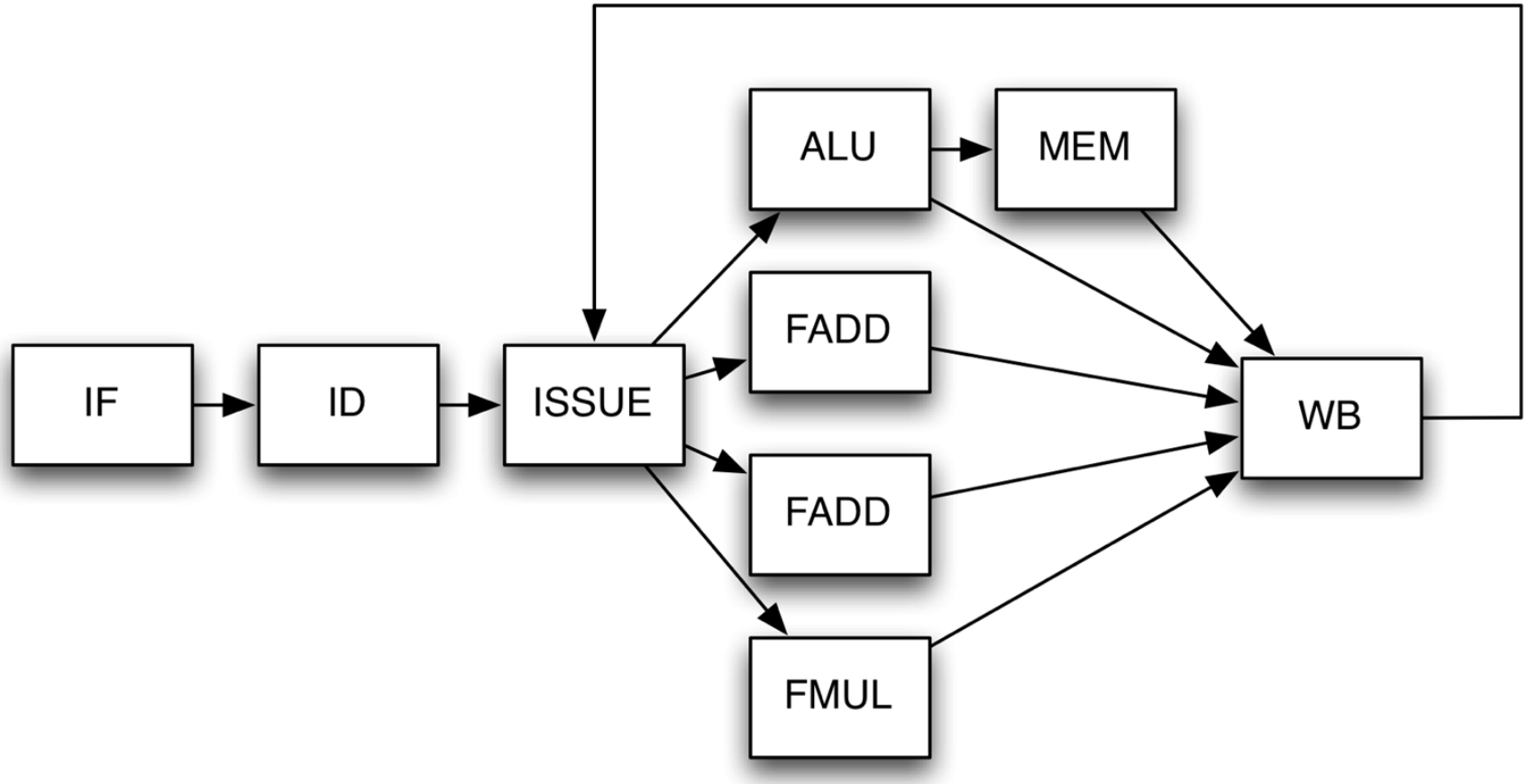
If the previous table was not correct, please, write the right one and specify the number, kind and latency for each unit.

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

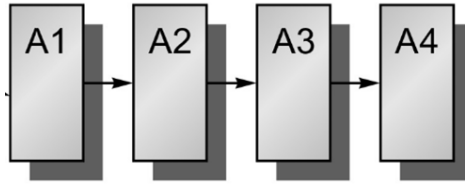
~~RAW F6 I1-I2~~
~~RAW F6 I1-I4~~
~~RAW F2 I2-I3~~
~~RAW F2 I2 I4~~
~~RAW F2 I2-I5~~
~~RAW F12 I4-I5~~
~~WAW F0 I3-I5~~

Exe: Complex Pipeline

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

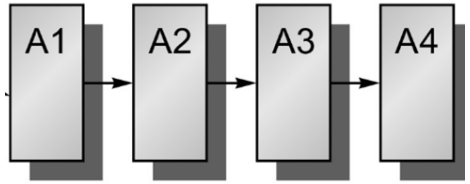


You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 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
- The target address for a branch is available in the FETCH stage

You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 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
- The target address for a branch is available in the FETCH stage

Conflicts/Dependencies

Assembly Code:

```
I1:  FOR:  ld    $f2, VB($r6)
I2:        fadd  $f3, $f2, $f6
I3:        st    $f3, VA($r7)
I4:        ld    $f3, VC($r6)
I5:        st    $f3, VC($r7)
I6:        fadd  $f4,$f4,$f3
I7:        addi  $r6, $r6, 4
I8:        addi  $r7, $r7, 4
I9:        blt   $r7, $r8, FOR
```

Conflicts/Dependencies

Assembly Code:

```
I1:  FOR:  ld   $f2, VB($r6)
I2:        fadd $f3, $f2, $f6
I3:        st   $f3, VA($r7)
I4:        ld   $f3, VC($r6)
I5:        st   $f3, VC($r7)
I6:        fadd $f4, $f4, $f3
I7:        addi $r6, $r6, 4
I8:        addi $r7, $r7, 4
I9:        blt  $r7, $r8, FOR
```

RAW F2 I1-I2
RAW F3 I2-I3
RAW F3 I4-I5
RAW F3 I4-I6
RAW R7 I8-I9

Conflicts/Dependencies

Assembly Code:

```
I1:  FOR:  ld   $f2, VB($r6)
I2:        fadd $f3, $f2, $f6
I3:        st   $f3, VA($r7)
I4:        ld   $f3, VC($r6)
I5:        st   $f3, VC($r7)
I6:        fadd $f4, $f4, $f3
I7:        addi $r6, $r6, 4
I8:        addi $r7, $r7, 4
I9:        blt  $r7, $r8, FOR
```

RAW F2 I1-I2
RAW F3 I2-I3
RAW F3 I4-I5
RAW F3 I4-I6
RAW R7 I8-I9
WAR R7 I8-I5
WAR R7 I8-I3
WAR R6 I7-I1
WAR R6 I7-I4
WAW F3 I2-I4
WAR F3 I3-I4

Conflicts/Dependencies

Assembly Code:

```
I1:  FOR:  ld   $f2, VB($r6)
I2:        fadd $f3, $f2, $f6
I3:        st   $f3, VA($r7)
I4:        ld   $f3, VC($r6)
I5:        st   $f3, VC($r7)
I6:        fadd $f4,$f4,$f3
I7:        addi $r6, $r6, 4
I8:        addi $r7, $r7, 4
I9:        blt  $r7, $r8, FOR
```

RAW F2 I1-I2
RAW F3 I2-I3
RAW F3 I4-I5
RAW F3 I4-I6
RAW R7 I8-I9
WAR R7 I8-I5
WAR R7 I8-I3
WAR R6 I7-I1
WAR R6 I7-I4
WAW F3 I2-I4
WAR F3 I3-I4
CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)																													
2	fadd $f3, $f2, $f6																													RAW I1 - I2 F2
3	st $f3, VA($r7)																													RAW I2 - I3 F3
4	ld $f3, VC($r6)																													WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)																													RAW I4 - I5 F3
6	fadd $f4,$f4,$f3																													RAW I4 - I6 F3
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6																													RAW I1 - I2 F2
3	st $f3, VA($r7)																													RAW I2 - I3 F3
4	ld $f3, VC($r6)																													WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)																													RAW I4 - I5 F3
6	fadd $f4,$f4,$f3																													RAW I4 - I6 F3
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)																													RAW I2 - I3 F3
4	ld $f3, VC($r6)																													WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)																													RAW I4 - I5 F3
6	fadd $f4,$f4,$f3																													RAW I4 - I6 F3
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS	IS	IS	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS	IS	IS	IS	IS	IS	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)																													WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)																													RAW I4 - I5 F3
6	fadd $f4,$f4,$f3																													RAW I4 - I6 F3
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS s	IS s	IS s	IS s	IS s	IS s	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D s	D s	D s	D s	D s	D s	D s	D s	D s	D	IS	E1	E2	E3	W										~~WAR I3- I4 F3~~ ~~WAW I2- I4 F3~~
5	st $f3, VC($r7)																													RAW I4 - I5 F3
6	fadd $f4,$f4,$f3																													RAW I4 - I6 F3
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS	IS	IS	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS	IS	IS	IS	IS	IS	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D	D	D	D	D	D	D	D	D	D	IS	E1	E2	E3	W										~~WAR I3- I4 F3~~ ~~WAW I2- I4 F3~~
5	st $f3, VC($r7)					F	F	F	F	F	F	F	F	F	F	D	IS	IS	IS	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$f3																													RAW I4 - I6 F3
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS s	IS s	IS s	IS s	IS s	IS s	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D s	D s	D s	D s	D s	D s	D s	D s	D s	D	IS	E1	E2	E3	W										~~WAR I3 -I4 F3~~ ~~WAW I2 -I4 F3~~
5	st $f3, VC($r7)					F s	F s	F s	F s	F s	F s	F s	F s	F s	F	D	IS s	IS s	IS s	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$f3															F	D	IS s	IS s	IS s	IS	E1	E2	E3	W					~~RAW I4 - I6 F3~~
7	addi $r6, $r6, 4																													WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS s	IS s	IS s	IS s	IS s	IS s	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D s	D s	D s	D s	D s	D s	D s	D s	D s	D	IS	E1	E2	E3	W										~~WAR I3 -I4 F3~~ ~~WAW I2 -I4 F3~~
5	st $f3, VC($r7)					F s	F s	F s	F s	F s	F s	F s	F s	F s	F	D	IS s	IS s	IS s	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$f3															F	D	IS s	IS s	IS s	IS	E1	E2	E3	W					~~RAW I4 - I6 F3~~
7	addi $r6, $r6, 4																F	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				~~WAR I1 -I7 r6~~ ~~WAR I4 -I7 r6~~
8	addi $r7, $r7, 4																													WAR I3-I8 r7, WAR I5-I8 r7,
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS s	IS s	IS s	IS s	IS s	IS s	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D s	D s	D s	D s	D s	D s	D s	D s	D s	D	IS	E1	E2	E3	W										~~WAR I3 -I4 F3~~ ~~WAW I2 -I4 F3~~
5	st $f3, VC($r7)					F s	F s	F s	F s	F s	F s	F s	F s	F s	F	D	IS s	IS s	IS s	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$f3															F	D	IS s	IS s	IS s	IS	E1	E2	E3	W					~~RAW I4 - I6 F3~~
7	addi $r6, $r6, 4																F	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				~~WAR I1 -I7 r6~~ ~~WAR I4 -I7 r6~~
8	addi $r7, $r7, 4																	F s	F	D s	D s	D s	D	IS s	IS	E1	W			~~WAR I3 -I8 r7,~~ ~~WAR I5 -I8 r7,~~ Struct ALU
9	blt $r7, $r8, FOR																													RAW R7 I8-I9
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS s	IS s	IS s	IS s	IS s	IS s	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D s	D s	D s	D s	D s	D s	D s	D s	D s	D	IS	E1	E2	E3	W										~~WAR I3 -I4 F3~~ ~~WAW I2 -I4 F3~~
5	st $f3, VC($r7)					F s	F s	F s	F s	F s	F s	F s	F s	F s	F	D	IS s	IS s	IS s	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$f3															F	D	IS s	IS s	IS s	IS	E1	E2	E3	W					~~RAW I4 - I6 F3~~
7	addi $r6, $r6, 4																F	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				~~WAR I1 -I7 r6~~ ~~WAR I4 -I7 r6~~
8	addi $r7, $r7, 4																	F s	F	D s	D s	D s	D	IS s	IS	E1	W			~~WAR I3 -I8 r7,~~ ~~WAR I5 -I8 r7,~~ ~~Struct ALU~~
9	blt $r7, $r8, FOR																			F s	F s	F s	F	D	IS s	IS s	IS	E1	W	~~RAW R7 I8 -I9~~
10	(New Loop Iteration)																													CNTRL

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS s	IS s	IS s	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS s	IS s	IS s	IS s	IS s	IS s	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D s	D s	D s	D s	D s	D s	D s	D s	D s	D	IS	E1	E2	E3	W										~~WAR I3 -I4 F3~~ ~~WAW I2 -I4 F3~~
5	st $f3, VC($r7)					F s	F s	F s	F s	F s	F s	F s	F s	F s	F	D	IS s	IS s	IS s	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$f3															F	D	IS s	IS s	IS s	IS	E1	E2	E3	W					~~RAW I4 - I6 F3~~
7	addi $r6, $r6, 4																F	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				~~WAR I1 -I7 r6~~ ~~WAR I4 -I7 r6~~
8	addi $r7, $r7, 4																	F s	F	D s	D s	D s	D	IS s	IS	E1	W			~~WAR I3-I8 r7,~~ ~~WAR I5-I8 r7,~~ ~~Struct ALU~~
9	blt $r7, $r8, FOR																			F s	F s	F s	F	D	IS s	IS s	IS	E1	W	~~RAW R7 I8-I9~~
10	(New Loop Iteration)																							F s	F s	F s	F s	F	D	CNTRL ~~CNTRL~~

Pipeline Schema

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)	F	D	IS	E1	E2	E3	W																						
2	fadd $f3, $f2, $f6		F	D	IS	IS	IS	IS	E1	E2	E3	W																		~~RAW I1 - I2 F2~~
3	st $f3, VA($r7)			F	D	IS	IS	IS	IS	IS	IS	IS	E1	E2	E3	W														~~RAW I2 - I3 F3~~
4	ld $f3, VC($r6)				F	D	D	D	D	D	D	D	D	D	D	IS	E1	E2	E3	W										~~WAR I3 -I4 F3~~ ~~WAW I2 -I4 F3~~
5	st $f3, VC($r7)					F	F	F	F	F	F	F	F	F	F	D	IS	IS	IS	IS	E1	E2	E3	W						~~RAW I4 - I5 F3~~
6	fadd $f4,$f4,$6															F	D	IS	IS	IS	I	E1	E2	E	W					~~RAW I4 - I6 F3~~
7	addi $r6, $r6,4																I	D	D	IS	D	IS	E1	E	F1	W				~~WAR I1 -I7 r6~~ ~~WAR I4 -I7 r6~~
8	addi $r7, $r7, 4																	F	F	D	D	D	D	IS	IS	E1	W			~~WAR I3-I8 r7,~~ ~~WAR I5-I8 r7,~~ ~~Struct ALU~~
9	blt $r7, $r8, FOR																			F	F	F	F	D	IS	IS	IS	E1	W	~~RAW R7 I8-I9~~
10	(New Loop Iteration)																							F	F	F	F	F	D	CNTRL

What if a Static Branch Prediction?

Recall: Static Branch Prediction Techniques

Branch Always Not Taken (Predicted-Not-Taken)

Branch Always Taken (Predicted-Taken)

Backward Taken Forward Not Taken (BTFNT)

Profile-Driven Prediction

Delayed Branch

No Branch Prediction

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS S	IS S	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS S	IS S	IS S	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D S	D	IS S	IS S	IS	E1	E1 S	E1 S	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F S	F	D S	D S	D S	D	IS S	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F S	F S	F S	F	D	IS S	IS S	IS	E1	W	RAW R7 I8-I9
10	(Following Instruction)							F S	F S	F S	F S	F	D	CNTRL

Recall: Static Branch Prediction Techniques

Branch Always Not Taken (Predicted-Not-Taken)

Branch Always Taken (Predicted-Taken)

Backward Taken Forward Not Taken (BTFNT)

Profile-Driven Prediction

Delayed Branch

No Branch Prediction

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS S	IS S	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS S	IS S	IS S	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D S	D	IS S	IS S	IS	E1	E1 S	E1 S	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F S	F	D S	D S	D S	D	IS S	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F S	F S	F S	F	D	IS S	IS S	IS	E1	W	RAW R7 I8-I9
10	(Following Instruction)							F S	F S	F S	F S	F	D	CNTRL

Static Branch Prediction: NT

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS s	IS s	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS s	IS s	IS s	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F s	F	D s	D s	D s	D	IS s	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F s	F s	F s	F	D	IS s	IS s	IS	E1	W	RAW R7 I8-I9
10	(Following Instruction)							F	D	IS s	IS s	IS flush?	E1 flush ?	CNTRL

Recall: Static Branch Prediction Techniques

Branch Always Not Taken (Predicted-Not-Taken)

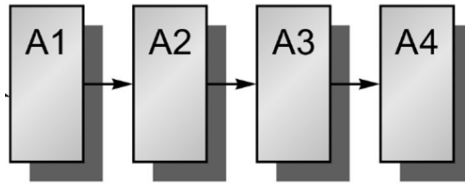
Branch Always Taken (Predicted-Taken)

Backward Taken Forward Not Taken (BTFNT)

Profile-Driven Prediction

Delayed Branch

You can assume that



- All functional units are pipelined
- ALU operations take 1 cycle
- Memory operations take 3 cycles (includes time in ALU)
- Floating-point add instructions take 3 cycles
- Floating-point multiply instructions take 5 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
- The target address for a branch is available in the FETCH stage

No Branch Prediction

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS S	IS S	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS S	IS S	IS S	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D S	D	IS S	IS S	IS	E1	E1 S	E1 S	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F S	F	D S	D S	D S	D	IS S	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F S	F S	F S	F	D	IS S	IS S	IS	E1	W	RAW R7 I8-I9
10	(Following Instruction)							F S	F S	F S	F S	F	D	CNTRL

Static Branch Prediction: T

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS s	IS s	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS s	IS s	IS s	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F s	F	D s	D s	D s	D	IS s	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F s	F s	F s	F	D	IS s	IS s	IS	E1	W	RAW R7 I8-I9
10	(I1 rexec)							F	D	IS s	IS s	IS flush?	E1 flus h?	CNTRL, RAW I7 - I10 R6

Recall: Static Branch Prediction Techniques

Branch Always Not Taken (Predicted-Not-Taken)

Branch Always Taken (Predicted-Taken)

Backward Taken Forward Not Taken (BTFNT)

Profile-Driven Prediction

Delayed Branch

No Branch Prediction

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS S	IS S	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS S	IS S	IS S	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D S	D	IS S	IS S	IS	E1	E1 S	E1 S	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F S	F	D S	D S	D S	D	IS S	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F S	F S	F S	F	D	IS S	IS S	IS	E1	W	RAW R7 I8-I9
10	(Following Instruction)							F S	F S	F S	F S	F	D	CNTRL

Static Branch Prediction: BTFNT

ALU OP: 1 cycle
MEM OP: 3 cycles
FP ADD: 3 cycles
FP MULT: 5 cycles

	Instruction	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	Notes
1	FOR: ld $f2,VB($r6)													
2	fadd $f3, $f2, $f6													RAW I1 - I2 F2
3	st $f3, VA($r7)													RAW I2 - I3 F3
4	ld $f3, VC($r6)	E2	E3	W										WAR I3-I4 F3 WAW I2-I4 F3
5	st $f3, VC($r7)	IS s	IS s	IS	E1	E2	E3	W						RAW I4 - I5 F3
6	fadd $f4,$f4,$f3	IS s	IS s	IS s	IS	E1	E2	E3	W					RAW I4 - I6 F3
7	addi $r6, $r6, 4	D s	D	IS s	IS s	IS	E1	E1 s	E1 s	W				WAR I1-I7 r6 WAR I4-I7 r6
8	addi $r7, $r7, 4	F s	F	D s	D s	D s	D	IS s	IS	E1	W			WAR I3-I8 r7, WAR I5-I8 r7, Struct ALU
9	blt $r7, $r8, FOR			F s	F s	F s	F	D	IS s	IS s	IS	E1	W	RAW R7 I8-I9
10	(I1 rexec)							F	D	IS s	IS s	IS flush?	E1 flus h?	CNTRL, RAW I7 - I10 R6

Recall:
Exe VLIW: performanc

```
FOR:  ld    $f2, VB($r6)
      fadd  $f3, $f2, $f6
      st    $f3, VA($r7)
      ld    $f3, VC($r6)
      st    $f3, VC($r7)
      fadd  $f4,$f4,$f3
      addi  $r6, $r6, 4
      addi  $r7, $r7, 4
      blt   $r7, $r8, FOR
```

	Integer ALU(1/2 b)	Memory Unit(3cc)	FPU(3cc)
C1		ld $f2, VB($r6)	
C2			
C3			
C4			fadd $f3, $f2, $f6
C5			
C6			
C7		st $f3, VA($r7)	
C8	addi $r6, $r6, 4	ld $f3, VC($r6)	
C9			
C10			
C11	addi $r7, $r7, 4	st $f3, VA($r7)	fadd $f4, $f4 , $f3
C12			
C13	blt $r7, $r8, FOR		
C14	(br delay slot)		
C15			

FPops/cycle=

=2 fadds / 14 cycles =

= 0.143

Thanks for your attention

Davide Conficconi <davide.conficconi@polimi.it>

Acknowledgements

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
- “Digital Design and Computer Architecture” Harris and Harris
- «On the Role of Reconfigurable Systems in Domain-Specific Computing»
- Elsevier Inc. online materials
- Papers/news cited in this lecture