POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

POLITECNICO MILANO 1863 NECST laboratory

Exercise Session 4

Scheduling: Static @ VLIW, Dynamic @ Scoreboard, Compiler @ Int Pipeline

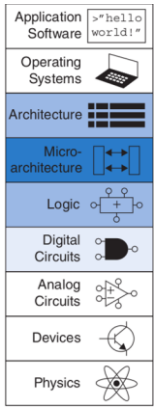
Advanced Computer Architectures

2nd April 2025

Davide Conficconi ‹davide.conficconi@polimi.it›

POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

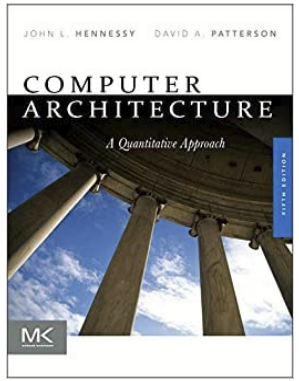
Recall: Material (EVERYTHING OPTIONAL)



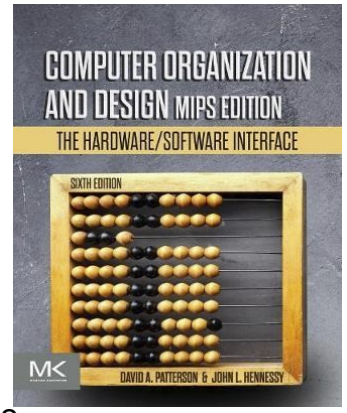
https://webeep.polimi.it/course/view.php?id=14754

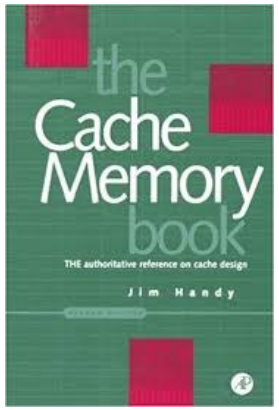
https://tinyurl.com/aca-grid25

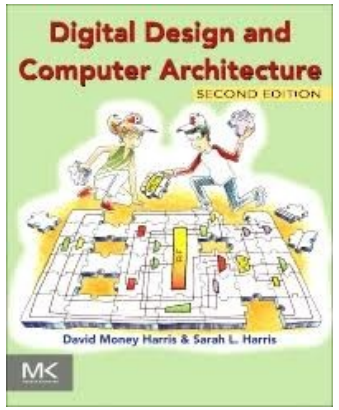
Textbook: Hennessy and Patterson, Computer Architecture: A Quantitative Approach

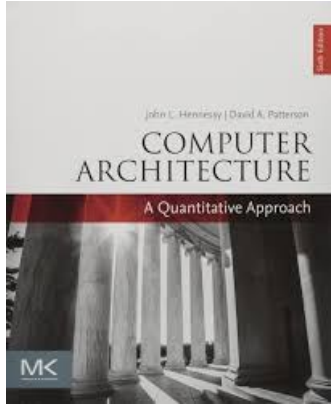


Other Interesting Reference



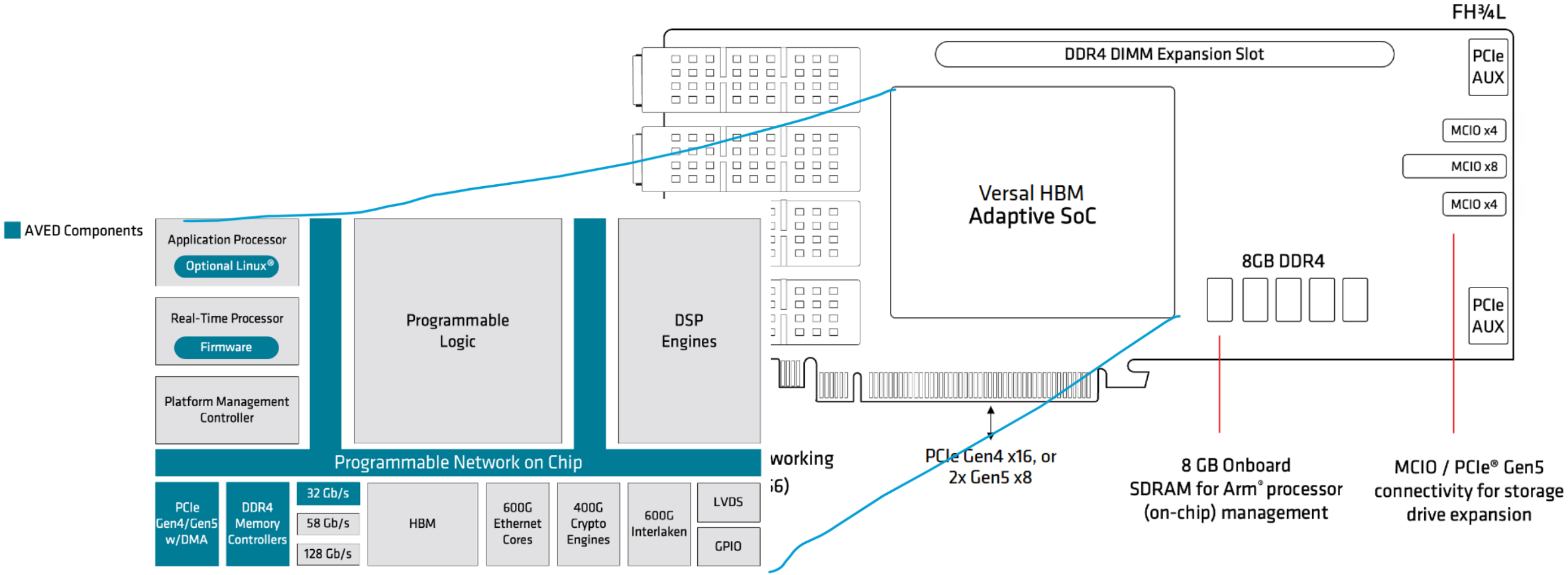




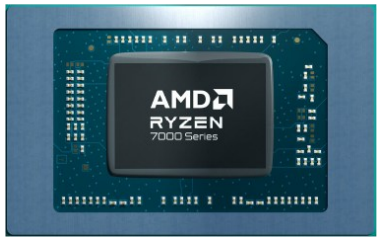


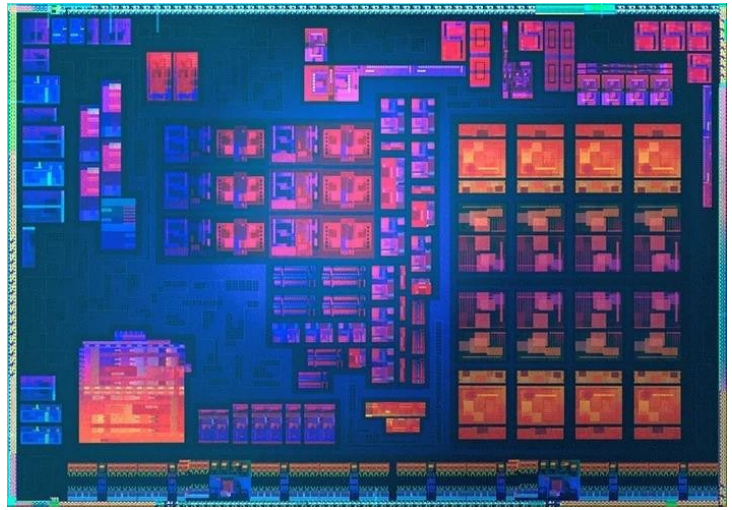
News from the outer world: AMD Alveo V80 Versal SoC

https://www.amd.com/en/products/accelerators/alveo/v80.html



News from the outer world: AMD Ryzen Phoenix' SoC: 7040



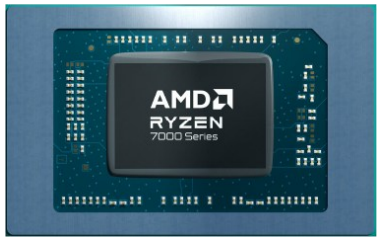


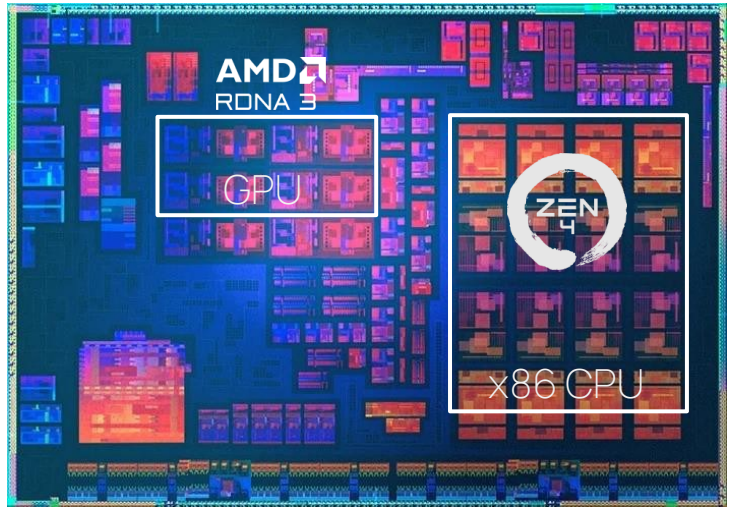
Phoenix
floorplan

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10592049; https://github.com/Xilinx/mlir-aie/tree/main/programming guide

AMD Ryzen 7040, HotChips-35 Conference, August 2023

News from the outer world: AMD Ryzen Phoenix' SoC: 7040



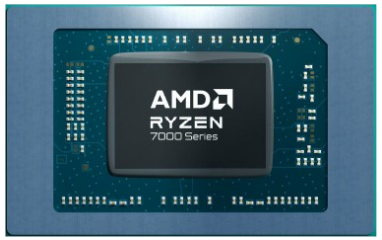


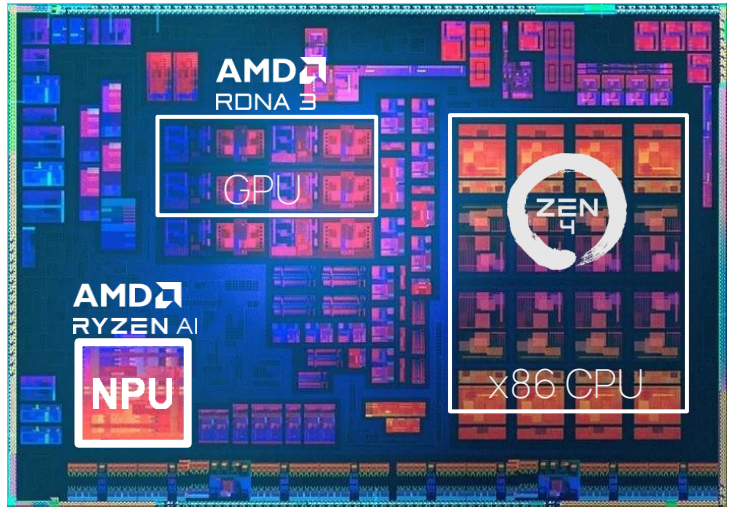
Phoenix floorplan

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10592049; https://github.com/Xilinx/mlir-aie/tree/main/programming guide

AMD Ryzen 7040, HotChips-35 Conference, August 2023

News from the outer world: AMD Ryzen Phoenix' SoC: 7040



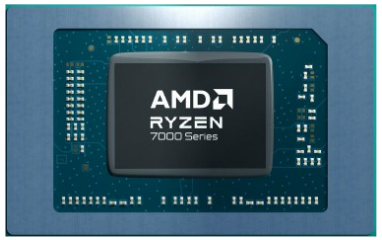


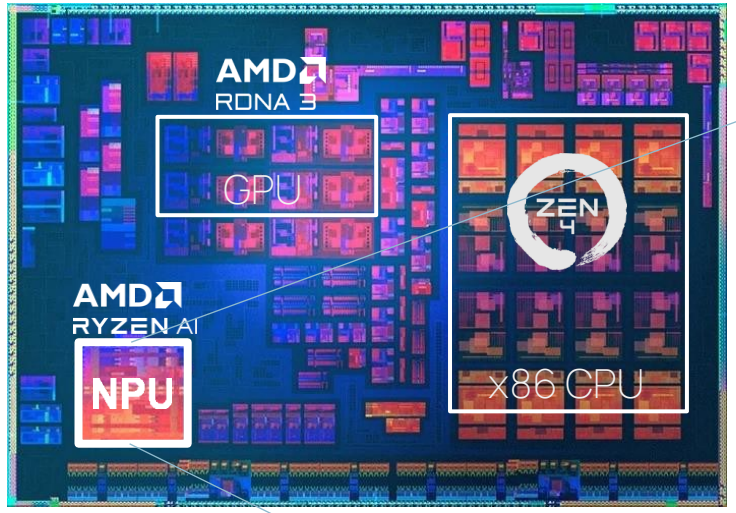
Phoenix floorplan

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10592049; https://github.com/Xilinx/mlir-aie/tree/main/programming guide

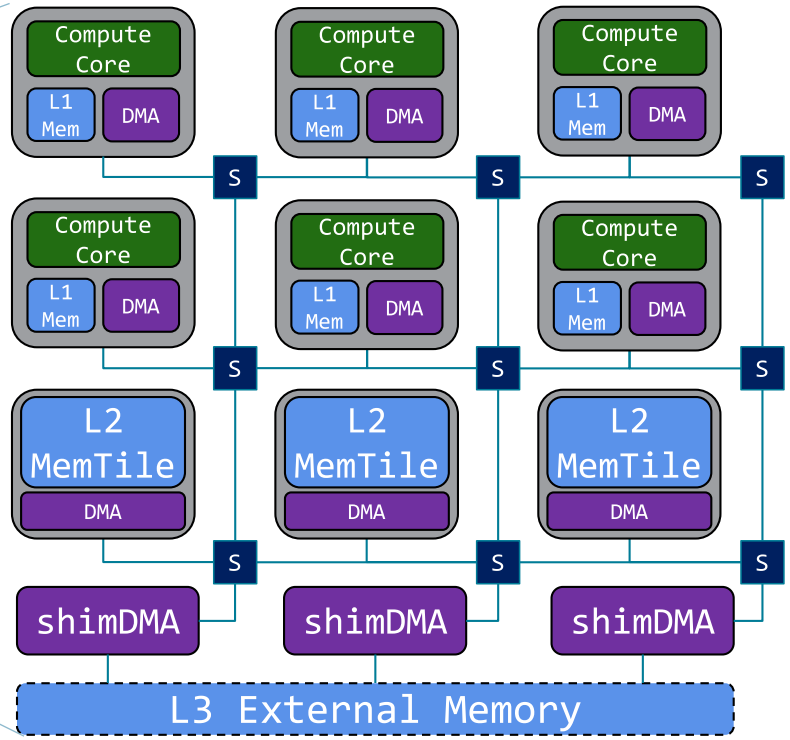
AMD Ryzen 7040, HotChips-35 Conference, August 2023

News from the outer world: AMD Ryzen Phoenix' SoC: 7040





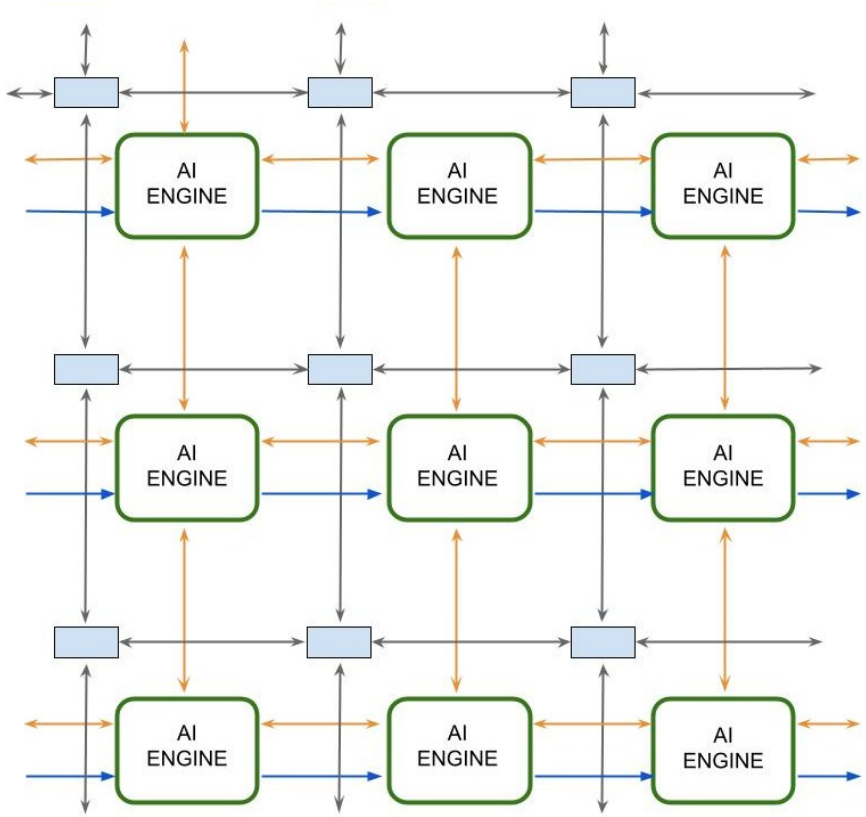
Phoenix floorplan

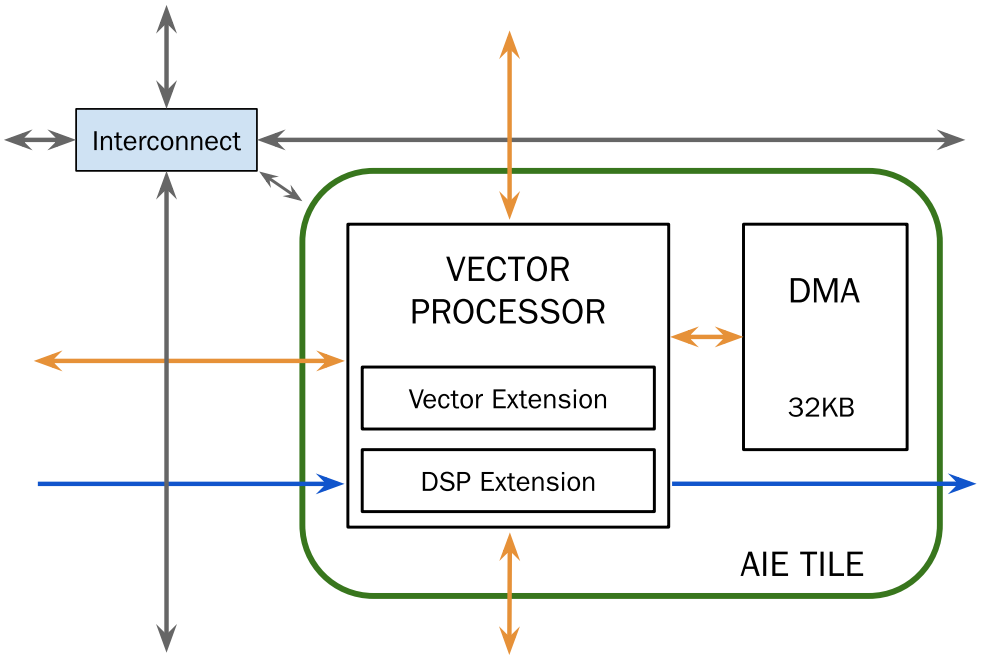


https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10592049; https://github.com/Xilinx/mlir-aie/tree/main/programming guide

AMD Ryzen 7040, HotChips-35 Conference, August 2023

News from the outer world: AI Engine Compute Accelerator





The AI Engine is a **7-way VLIW processor**[1]

scalar	move	move	load	load	store	vector

Program memory is limited to **1024 instructions**[1]

[1] https://docs.amd.com/r/en-US/am009-versal-ai-engine

SUPERCALIFRAGILISTICHESPIRALIDOSO (AKA VLIW)

Recall: Scheduling Idea

To improve ILP (instruction-level parallelism) dependences must be detected and solved, and instructions must be ordered (scheduled) so as to achieve highest parallelism of execution compatible with available resources.

Two properties must be preserved

- Exception behavior: Preserving exception behavior means that any changes in the ordering of instruction execution must not change how exceptions are raised in the program.
- Data flow: Actual flow of data values among instructions that produces the correct results and consumes them.

Recall: Instruction Level Parallelism

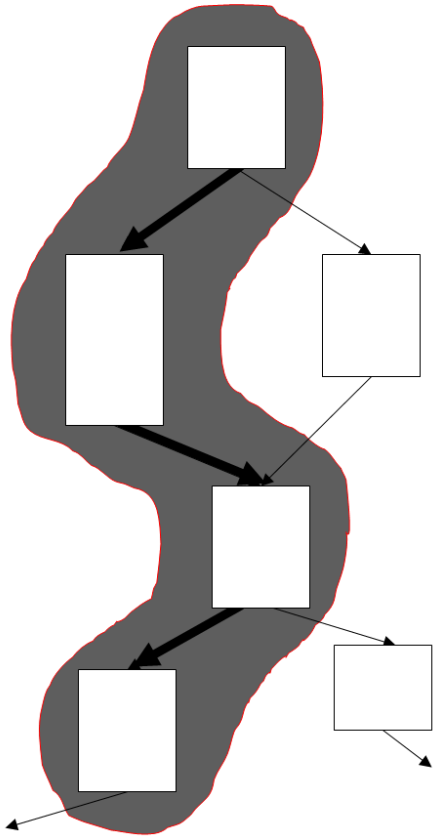
Two strategies to support ILP:

- Dynamic Scheduling: Depend on the hardware to locate parallelism
- Static Scheduling: Rely on software for identifying potential parallelism

Hardware intensive approaches dominate desktop and server markets

Recall: Static Scheduling

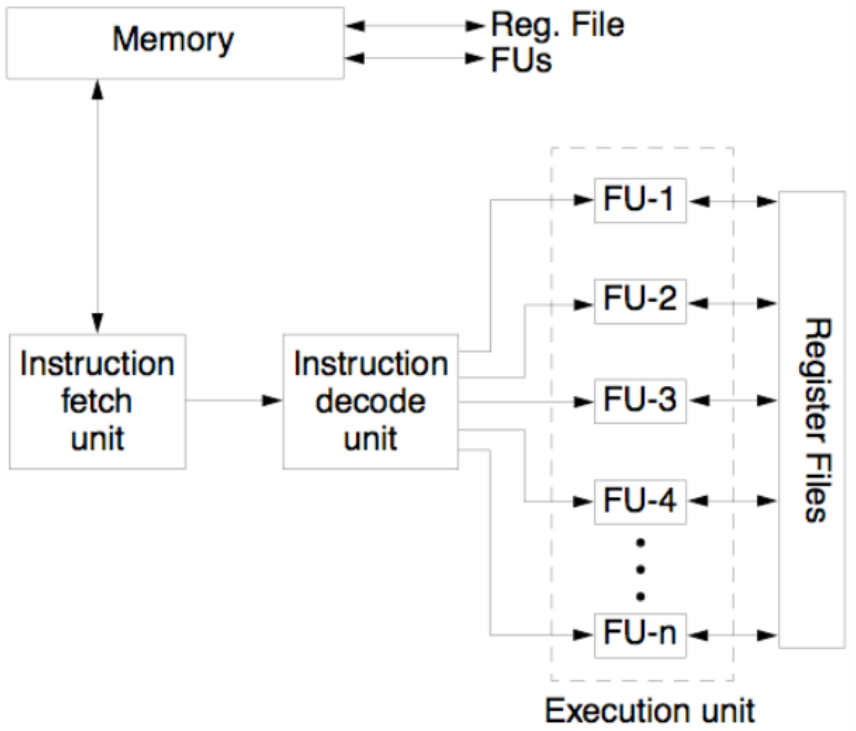
Compilers can use sophisticated algorithms for code scheduling to exploit ILP (Instruction Level Parallelism) in a **basic block** – a straight-line code sequence with no branches in except to the entry and no branches out except at the exit

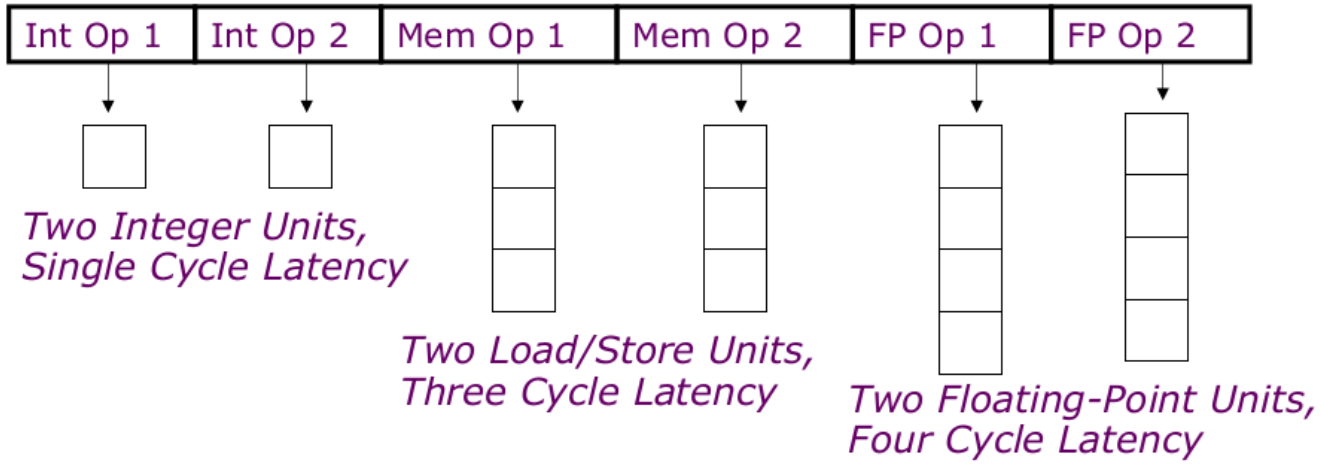


Static detection and resolution of dependences (static scheduling): accomplished by the **compiler** ⇨ dependences are avoided by **code reordering**. Output of the compiler: reordered into dependency-free code.

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

Recall VLIW and Static Scheduling





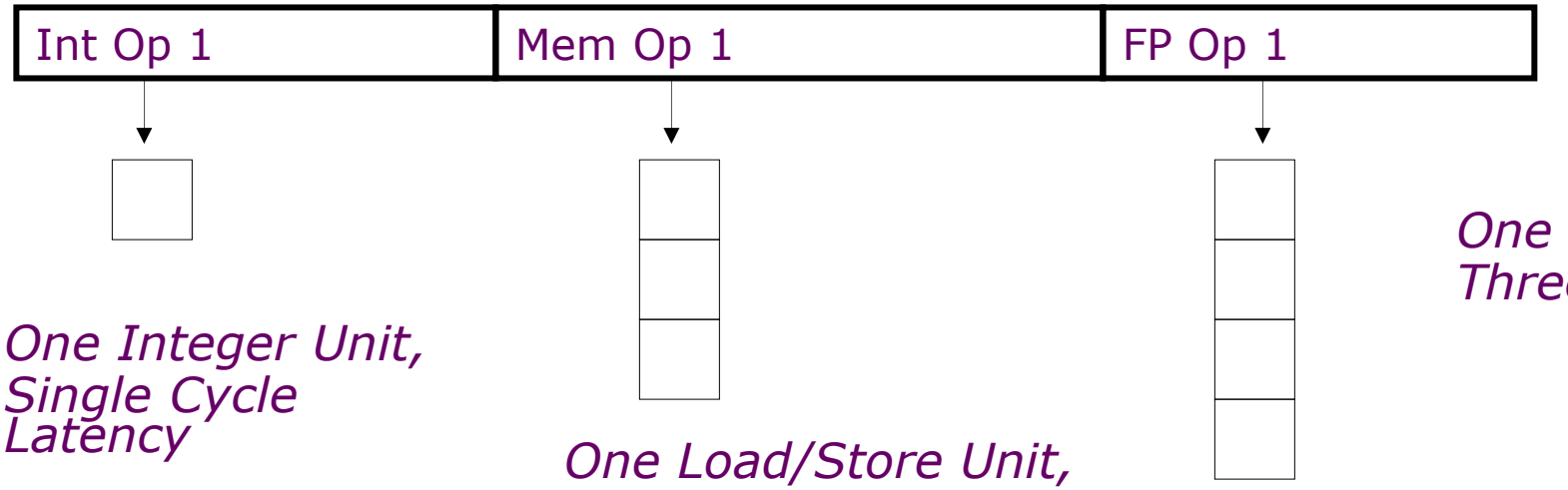
Static Scheduling: Rely on software for identifying potential parallelism
(example of List Based Scheduling with ASAP)

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

Exe VLIW: schedule

Exe 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 to **next Integer/FP** and **2** cycle latency to **next Branch**
- **Memory** Unit with **3** cycle latency
- **Floating** Point Unit with **3** cycle latency (each **FPU** can complete one add or one multiply per clock cycle)
- Branch completed with **1 cycle delay slot** (branch solved in ID stage)



One Integer Unit, Single Cycle Latency

One Load/Store Unit, Three Cycle Latency

One Floating-Point Units, Three Cycle Latency

Recall Delayed Branch

(Static Branch Prediction Techniques)

- The job of the compiler is to make the instruction placed in the branch delay slot valid and useful.
- There are three ways in which the branch delay slot can be scheduled:
  1. From before
  2. From target
  3. From fall-through

Exe VLIW: 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**
- **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: the code

Assembly Code:

```
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
```

Exe VLIW: 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
```

Exe VLIW: 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

Exe VLIW: 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

Exe VLIW: 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

Exe 2 VLIW: schedule

```
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
```

ALU 1 cc Integer, 2 cc Branch

MU 3 cc

FPU 3 cc

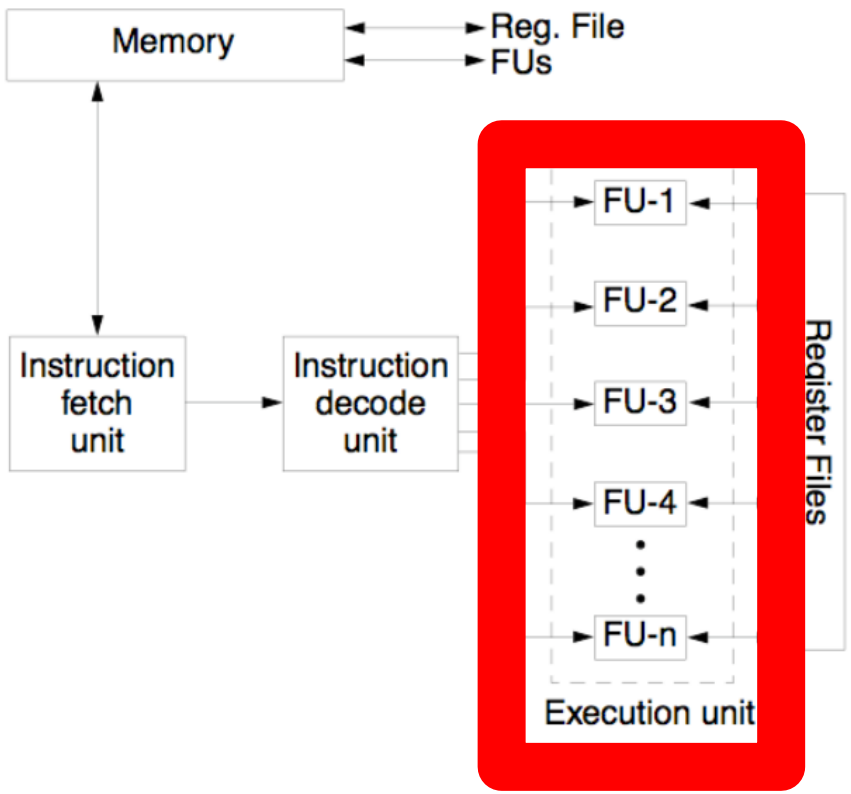
Exe 2 VLIW: schedule

```
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
```

ALU 1 cc Integer, 2 cc Branch

MU 3 cc

FPU 3 cc



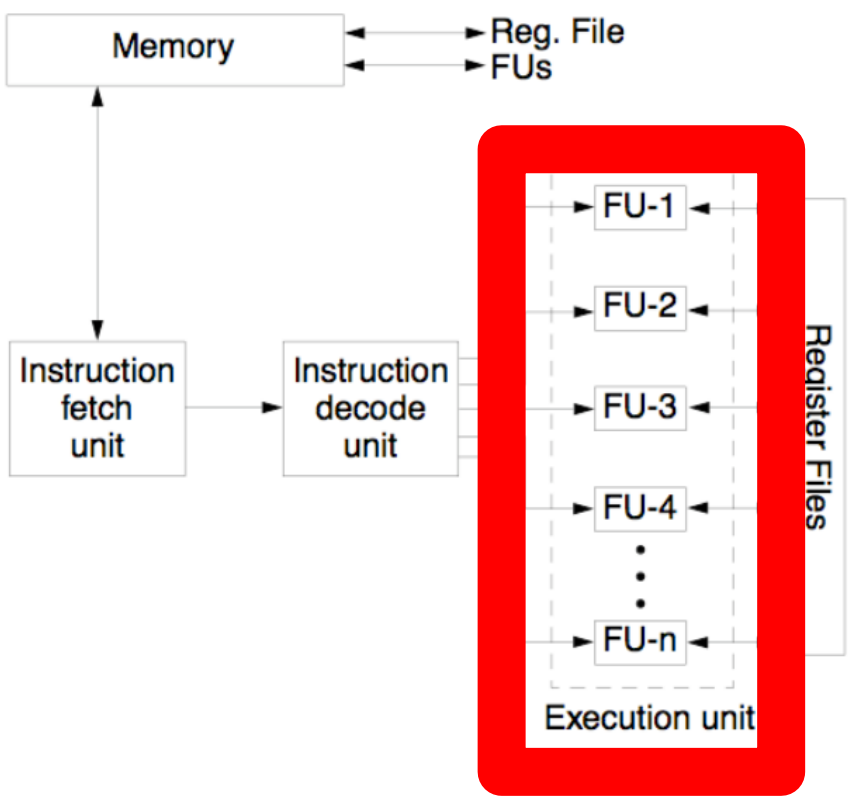
Exe 2 VLIW: schedule

```
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
```

ALU 1 cc Integer, 2 cc Branch

MU 3 cc

FPU 3 cc



- Schedule the instructions
- Calculate the FP ops / cycle

Exe VLIW: schedule

```
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			
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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			
C2			
C3			
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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) 1cc	
C2		2cc	
C3		3cc	
C4			
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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) 1cc	
C2		2cc	
C3		3cc	
C4			fadd $f3, $f2, $f6
C5			
C6			
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		1cc	fadd $f3, $f2, $f6
C5		2cc	
C6		3cc	
C7			
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		1cc	fadd $f3, $f2, $f6
C5		2cc	
C6		3cc	
C7		st $f3, VA($r7)	
C8			
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		ld $f3, VC($r6)	
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		ld $f3, VC($r6)	
C9			
C10			
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		ld $f3, VC($r6) 1cc	
C9		2cc	
C10		3cc	
C11			
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		ld $f3, VC($r6)	
C9			
C10			
C11		st $f3, VA($r7)	
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		ld $f3, VC($r6)	
C9			
C10			
C11		st $f3, VA($r7)	fadd $f4, $f4 , $f3
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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		st $f3, VA($r7)	fadd $f4, $f4 , $f3
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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
```

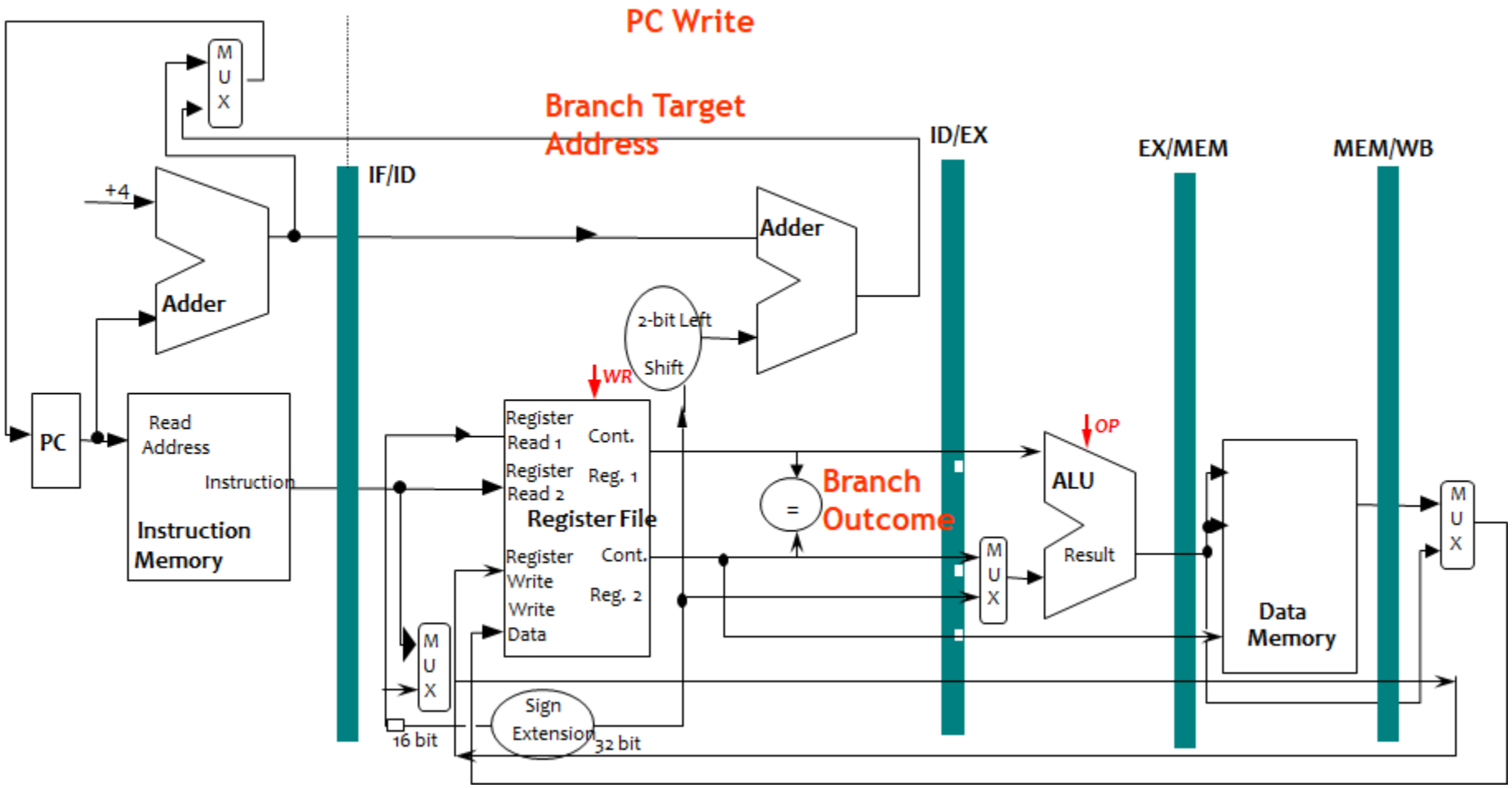
	Integer ALU(1/2 b)	Memory Unit(3cc)	FPU(3cc)
C1		ld $f2, VB($r6)	
C2			
C3			
C4			fadd $f3, $f2, $f6
C5			
C6			
C7		st $f3, VA($r7)	
C8	addi $r6, $r6, 4	ld $f3, VC($r6)	
C9			
C10			
C11	addi $r7, $r7, 4	st $f3, VA($r7)	fadd $f4, $f4 , $f3
C12			
C13			
C14			
C15			

Exe VLIW: schedule

```
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			
C14			
C15			

Recall: Early-evaluation PC



Exe VLIW: schedule

```
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			
C15			

Exe VLIW: schedule

```
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			

Exe VLIW: actual code

```
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	NOP	ld $f2, VB($r6)	NOP
C2	NOP	NOP	NOP
C3	NOP	NOP	NOP
C4	NOP	NOP	fadd $f3, $f2, $f6
C5	NOP	NOP	NOP
C6	NOP	NOP	NOP
C7	NOP	st $f3, VA($r7)	NOP
C8	addi $r6, $r6, 4	ld $f3, VC($r6)	NOP
C9	NOP	NOP	NOP
C10	NOP	NOP	NOP
C11	addi $r7, $r7, 4	st $f3, VA($r7)	fadd $f4, $f4 , $f3
C12	NOP	NOP	NOP
C13	blt $r7, $r8, FOR	NOP	NOP
C14	(br delay slot)	NOP	NOP
C15	NOP	NOP	NOP

Exe VLIW: schedule

```
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			

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
```

FPops/cycle=

	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			

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

Recall: Instruction Level Parallelism

Two strategies to support ILP:

- Dynamic Scheduling: Depend on the hardware to locate parallelism
- Static Scheduling: Rely on software for identifying potential parallelism

Hardware intensive approaches dominate desktop and server markets

Recall: Key Idea: dynamic scheduling

Problem:

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

Solution: allow instructions behind a stall to proceed

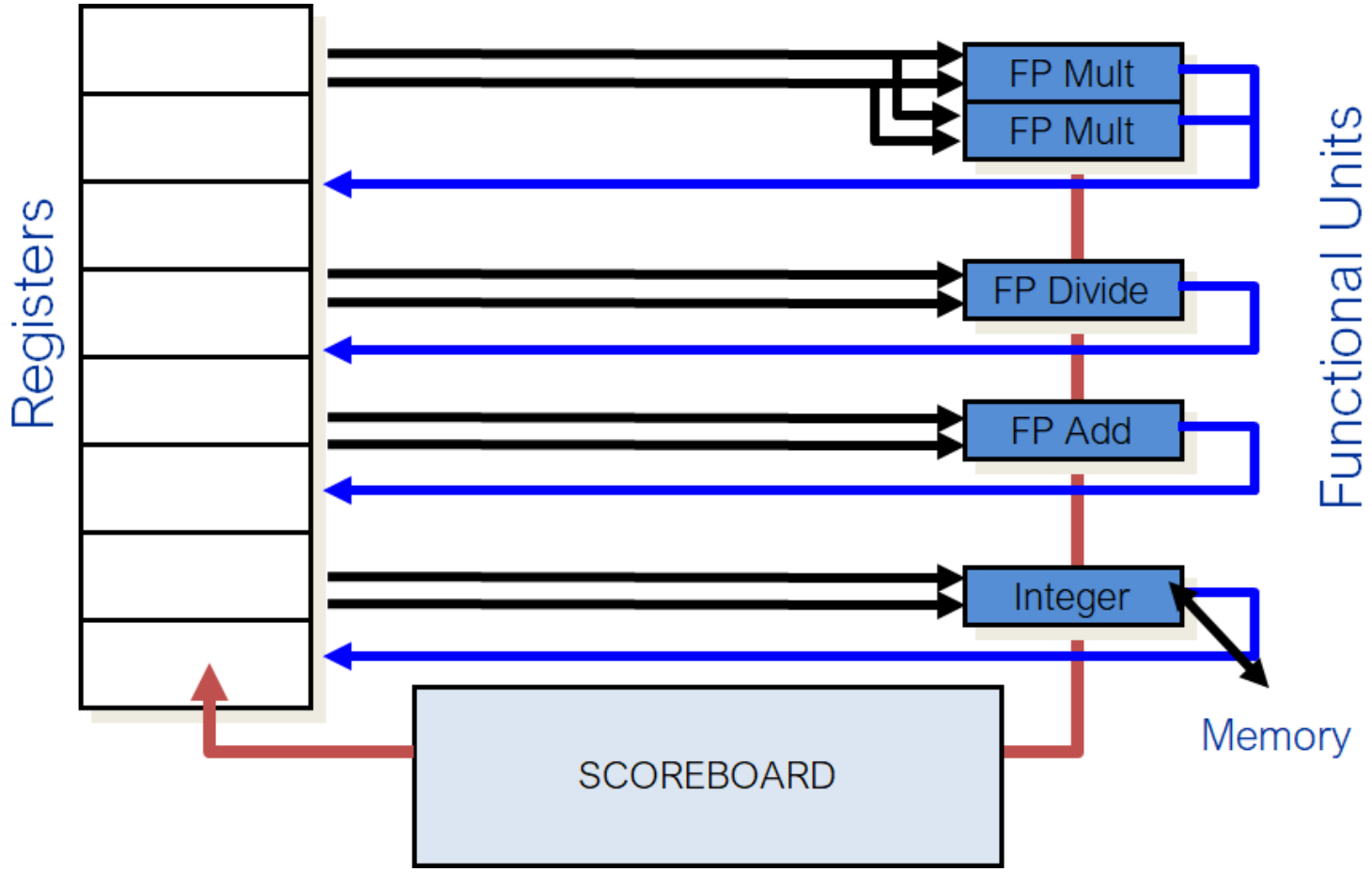
- HW rearranges the instruction execution to reduce stalls

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

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

First implemented in CDC6600 (1963)

Exe 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 Scoreboard: the Code

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

Exe 1.1 Scoreboard: Conflicts

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

Exe 1.1 Scoreboard: Conflicts

RAW F6 I1-I2

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

Exe 1.1 Scoreboard: Conflicts

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

RAW F6 I1-I2

RAW F6 I1-I4

Exe 1.1 Scoreboard: Conflicts

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

RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

Exe 1.1 Scoreboard: Conflicts

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

RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

Exe 1.1 Scoreboard: Conflicts

```
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

Exe 1.1 Scoreboard: Conflicts

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

RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

RAW F2 I2-I5

RAW F12 I4-I5

Exe 1.1 Scoreboard: Conflicts

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

RAW F6 I1-I2

RAW F6 I1-I4

RAW F2 I2-I3

RAW F2 I2-I4

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 1.2 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 1.2 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 1.2 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 1.2 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 1.2 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 1.2 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 1.2 Scoreboard: ▇ a configuration?

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

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

Exe 1.3 Scoreboard: if not correct, write right one

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

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

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

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

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

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

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

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

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

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

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

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

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

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

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

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

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

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

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

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

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

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

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