

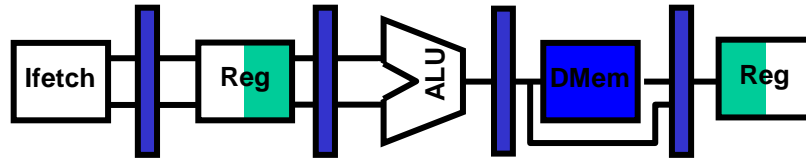
Tutorial exercise 1: Pipeline architecture and instruction set are coupled

Advanced Computer Architecture

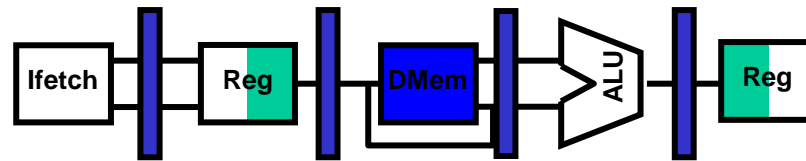
Paul H J Kelly

Let's fiddle with it....

- Here is the 5-stage DLX/MIPS pipeline from the textbook:



- Suppose we rearrange the pipeline
- Eg so the MEM stage comes before the ALU:



- Your job:
 - Have an opinion about whether this is a good idea**
 - Have a plan for finding out**

The proposed instruction set redesign

- For example, suppose R1 is an index into an array starting at address 100, and we need to multiply the R1th element by 123. On MIPS you could write:

```
LD R2, 100(R1)
```

```
MULI R3, R2, 123
```

- Here we load R2 with the element of the array, using displacement addressing. With the modified architecture, you would have to write

```
ADDI R2, R1, 100
```

```
MULI R3, (R2), 123
```

- Here, the first instruction does the displacement calculation explicitly, leaving a pointer to the array element in R2. The second instruction then reads the value from memory.

Where are the stalls?

<i>Clock:</i>	1	2	3	4	5	6	7	8	9
Instruction 1	IF	ID	MEM	EX	WB				
Instruction 2		IF	ID	MEM	EX	WB			
Instruction 3			IF	ID	MEM	EX	WB		
Instruction 4				IF	ID	MEM	EX	WB	
Instruction 5					IF	ID	MEM	EX	WB

- Question:

When will we suffer a pipeline stall?

- In the original pipeline we got a stall between a load and a use:

LD R1, R2(100)

ADD R3, R1, 1

Where are the stalls?

<i>Clock:</i>	1	2	3	4	5	6	7	8	9
Instruction 1	IF	ID	MEM	EX	WB				
Instruction 2		IF	ID	MEM	EX	WB			
Instruction 3			IF	ID	MEM	EX	WB		
Instruction 4				IF	ID	MEM	EX	WB	
Instruction 5					IF	ID	MEM	EX	WB

- Question:

When will we suffer a pipeline stall?

- Consider:

ADD R1, R2, R3

ADD R4, (R1), R5

<i>Clock:</i>	1	2	3	4	5	6	7	8	9
Instruction 1	IF	ID	MEM	EX	WB				
Instruction 2		IF	ID	MEM	EX	WB			
Instruction 3			IF	ID	MEM	EX	WB		
Instruction 4				IF	ID	MEM	EX	WB	
Instruction 5					IF	ID	MEM	EX	WB

- Question:

When will we suffer a pipeline stall?

- Consider:

ADD R1, R2, R3

ADD R4, (R1), R5

- The change also has consequences for register allocation?

<i>Clock:</i>	1	2	3	4	5	6	7	8	9
Instruction 1	IF	ID	MEM	EX	WB				
Instruction 2		IF	ID	MEM	EX	WB			
Instruction 3			IF	ID	MEM	EX	WB		
Instruction 4				IF	ID	MEM	EX	WB	
Instruction 5					IF	ID	MEM	EX	WB

- Consider:

ADD R1, R2, R3

some other instruction

ADD R4, (R1), R5

Is this design better than the original? Worse?

How would you find out?

- Impact on clock time?
- Percentage of loads & stores that use displacement addressing
 - since each of these would now involve an additional instruction to compute the effective address.
- Percentage of loads which load a value which is only used once by an ALU operation
 - since it is in this case that the new addressing mode saves us something.
- After the compilers have been changed, what percentage of instructions that calculate an address are immediately followed by a load instruction which uses it?
 - since this is the case where a stall would occur).
- Compiler needs to know where stalls might occur
 - to schedule instructions to avoid them

Conclusions?

- It depends
- On exactly how applications and the compiler use the instruction set
- Influences:
 - Number of instructions you need to execute
 - Number of registers needed
 - Number of stalls
- Influenced by:
 - Compiler's ability to schedule instructions
 - Compiler's effectiveness in selecting instructions
 - Application behaviour
- Need to build compiler, and evaluate across wide, diverse, representative suite of applications you (or your customers) care about