

Notes on tutorial exercise 1

Advanced Computer Architecture

Paul H J Kelly

Exercise 1.1. Instruction set issues in pipeline design

- New addressing mode for a MIPS-like architecture.
- Idea is to allow one of operands of an ALU instruction to be in memory
- To offset this increase in complexity, you restrict all memory addressing (in Load, Store and ALU instructions) to be register-indirect only (i.e. no displacement addressing)
- **Propose a change to the 5-stage MIPS pipeline (as shown in the notes) to accommodate this addressing mode. Draw the new pipeline and explain the changes.**

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.

- Notice that the proposed changes do not force us to have two ALU stages: arithmetic instructions need an ALU to do the arithmetic, but do not need to do any effective address calculation (this is why it was so important to disallow displacement addressing).

- However, in processing an arithmetic instruction with a memory operand, it is necessary to access memory *before* performing the arithmetic, so we must have the MEM cycle before the EX cycle.

- So it looks like a solution is simply to switch the MEM and EX stages in the MIPS pipeline:

<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

<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:
 - **Give an example showing how a pipeline stall can arise with this modified design**
- 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:
 - **Give an example showing how a pipeline stall can arise with this modified design**
- 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

- Consider:

ADD R1, R2, R3

some other instruction

ADD R4, (R1), R5

Issues in evaluating proposal

- **Is this design better than the original? Worse? How would you find out?**
 - Impact on clock time?
 - Percentage of loads and stores which 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 so that displacement addressing is calculated in a register, 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