CS4617 Computer Architecture

Lecture 8: Pipelining

Reference: Appendix C, Hennessy & Patterson

Reference: Hamacher *et al.*

Dr J Vaughan

October 6, 2014

Preliminaries

- Processor cycle = time to move instr 1 step down the pipeline
- Usually 1 clock cycle
- Goal: balance length of pipeline stages
- Perfect balance
  $\Longrightarrow$ Time per instruction$= \frac{\textit{Time per instruction unpipelined}}{\textit{number of stages}}$
- $\Longrightarrow$ increase instructions/sec = *throughput*
  Also decrease average number of clock cycles per instruction (CPI)
- Pipelining is not visible to the programmer

MIPS-64: 64-bit version of MIPS

- DADD: 64-bit version of ADD
- LD: 64-bit version of load
- 32 registers
- Reg0 $= 0$

MIPS-64 Operations (1)

- ALU ops: $reg_{dest} \leftarrow reg_{source}$ op reg_{source}
  DADD, DSUB, AND, OR
- Immediate: mnemonic suffix I
- Signed/unsigned arithmetic
- Unsigned operations do not generate overflow exceptions
  $\Longrightarrow$ same for 32-bit and 64-bit
  $\rightarrow$ DADDU, DSUBU, DADDIU

MIPS-64 Operations (2)

- Load/store reg_{source} (base reg) + immediate offset (16 bits)
- EA = (base reg) + sign extended offset
- Load $\implies$ dest reg
  LD: load 64-bit register contents
- Store $\implies$ source reg
  SD: store 64-bit register contents

MIPS-64 Operations (3)

- Conditional branches/jumps:
  consider only equality in these examples
- MIPS:
  Compare pair of registers
  Compare register to zero
- Branch destination: sign-extend offset and add to current PC

MIPS without pipelining

- 5 clock cycles at most
- Load/store word
- Branch
- Integer ALU ops

1. IF
    - $IR \leftarrow (m < (PC) >)$
    - $PC \leftarrow (PC) + 4$

MIPS without pipelining (continued)

2 ID

- Decode, read regs from reg file
- Test regs for equality as they are read, for a possible branch sign-extend the offset field in case it is needed
- Compute possible branch target address by adding the sign-extended offset to the incremented PC
- Possible to implement branch aggressively at the end of this stage by storing the branch target address into the PC if the condition test is true
- Decoding in parallel with reading regs is possible because reg specifiers are at a fixed location in a RISC architecture: *fixed-field decoding*
- Sign-extend immediate is also possible during decoding for the same reason

MIPS without pipelining (continued)

3 EX (execution/effective address cycle)
   - Depends on instruction type
   - Mem ref: ALU adds base and offset $\rightarrow$ ea
   - Reg-Reg ALU: ALU performs specified operation on Rs and Rt
   - Reg-Immed ALU: ALU performs specified operation on Rs and sign-extended immediate value
   - Load-store architecture $\implies$ ea and execution cycles can be combined because no instruction needs to calculate data address at same time as performing an operation on data

MIPS without pipelining (continued)

4. MEM
   - Use ea calculated in (3)
   - Load: $\leftarrow (M < ea >)$
   - Store: Reg $\rightarrow M < ea >$

MIPS without pipelining (continued)

5 WB

- Reg-Reg ALU or load:
  Reg file $\leftarrow$ result from memory or ALU
- Branch needs 2 cycles
- Store needs 4 cycles
- All other instructions need 5 cycles
- If branch frequency 12%, store frequency 10%, overall CPI is 4.54

5-stage pipeline

	Clock cycle								
Instr No	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
i	IF	ID	EX	MEM	WB				
i+1		IF	ID	EX	MEM	WB			
i+2			IF	ID	EX	MEM	WB		
i+3				IF	ID	EX	MEM	WB	
i+4					IF	ID	EX	MEM	WB

Table: Figure C.1: Simple RISC pipeline