Computer Architecture Exercises

Series 4 & 5

Adrian Wälchli

May 1, 2018

Today

Review of Series 4

Introduction to Series 5

Organisation

Review of Series 4

Sign Extension

- `immediate`-values are signed
- e.g. negative offset
- `immediate`-values are 16 bits long

$\Rightarrow$ sign extension needed for correct computations with 32-bit numbers

Logical and Bitwise Operations

Type	**Input**	**Output**
bitwise	`0xA3 & 0x3A`	`0x22`
logical	`0x0 && 0xEF`	`0x0` or `FALSE`
bitwise	`0xA3 | 0x3A`	`0xBB`
logical	`0x0 || 0xEF`	`0x0` or `FALSE`
bitwise	$\sim$`0xFE`	`0x01` when `char`
		`0xFF FF FF 01` = -255 when `int`
logical	`!0xFE`	`0x00` or `FALSE`

Hardware Access

Isolated I/O

CPU has specific instructions for communication with connected hardware. Also known as port-mapped I/O.

Memory Mapped I/O

Memory and registers of connected hardware are mapped to address space of CPU. Reading and writing is no different than read/write access to regular memory. See also Raspberry Pi exercises as an example.

`bne` instead of `beq`

Singlecycle

invert `zero` output of ALU
(see slide 31 in "Basic MIPS Architecture Review")

Multicycle

invert `zero` output of ALU
(see slide 42 in "Basic MIPS Architecture Review")

Pipeline Registers

- used to separate each stage in the pipeline
- value of instruction needs to be preserved across all four stages
- need to be wide enough to save all data of each stage
- e.g. ID/IF is 64 bit wide in order to hold the 32-bit instruction fetched from memory and the 32-bit PC-address.

Hazards

Structural Hazards

The same resource (e.g. memory) is accessed by two different instructions at the same time.

Data Hazards

Data (e.g. result of arithmetic operation) is accessed/used before it is ready.

Control Hazards

There is not enough information available to decide which instruction should be executed next (e.g. branch prediction).

Stall

- edge-triggered registers
- during same clock: first write, then read
- both edges of clock used
- $\Rightarrow$ stall **two** clocks for data hazards
  (see slide 15 of "Basic MIPS Pipelining Review")
- control hazards: stall **three** clocks because new instruction needs to be fetched too
  (see slide 19 of "Basic MIPS Pipelining Review")

Data Hazard

```
add $t0, $t5, $t4
lw  $s2, 0($t0)
sub $s3, $t0, $s2
sw  $t4, 4($s3)
```

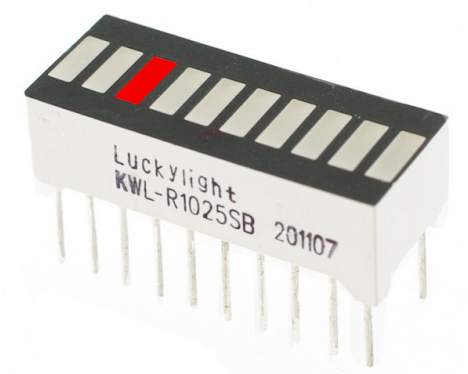
- `t0` `sub` and `lw` will access register before result of `add` is ready $\Rightarrow$ resolved by **forwarding**
- `s2` `sub` requires result before `lw` can load it $\Rightarrow$ leads to **stall**
- `s3` `sw` will access register before result of `sub` is ready $\Rightarrow$ resolved by **forwarding**

Introduction to Series 5

Programming Exercise

Paddle Ball

- extend your running light to a single player game
- digital version of “Paddle Ball”
- hit ball when it touches edge (+1 point)
- miss = buzzer sound (+0 points)
- speed increases
- display score at end of round



Bonus Task

Timer

- set a time using buttons
- position of light = time in minutes
- start the countdown with button
- blinking light shows remaining time and moves down
- plays sound when time runs out

Grading

This task is **not mandatory**, contributes to final grade.
Final score = written exam score + up to 10% of score

Exercise Grading

Theoretical- and programming parts are graded independently.

Theoretical Part

You need 60% of maximum score in **each** Series.

Programming Part

Each part is graded with 0, 0.5 or 1 point.
You need 4 of 5 points by the end of the semester.

Next Weeks

8. May	Pool session Series 5
15. May	Review Serie 5, Return Raspberry Pi, Repetition Series MIPS and C (not mandatory)
22. May	Solutions to Repetition Series (not mandatory)
29. May	Mock exam from previous years (not graded).
4. June	Exam