Single Cycle

Jonathan Windle

University of East Anglia

J.Windle@uea.ac.uk

June 10, 2017

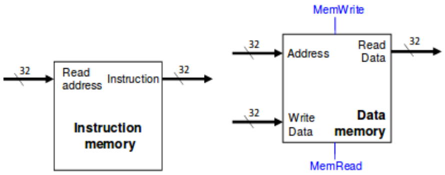
Overview I

State Machines

- Computers are state machines
- Built from combinational and state elements
- Combinational element outputs depend only on their inputs
    - e.g. gates, ALU
- State elements hole state i.e. have some internal storage
    - E.g. registers, instruction and data memories
    - Clocking methodology defines when data can be read and written
    - We will assume positive edge-triggered clocking.
- Edge-triggered clocking methodology allows for reading a state element and writing in the same clock cycle
- The clock needs to be long enough so that the input to the state element is stable before the positive clock edge starts.

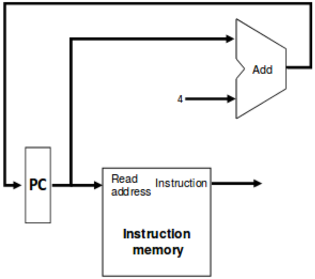
Instruction and Data Memories

- All buses are 32 bits
- Blue connectors represent control signals. MemWrite or MemRead should be asserted (logical high) depending whether we want to write or read from memory
- For simplicity, assume cannot write to instruction memory - program already there.



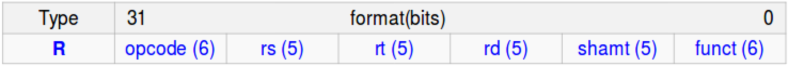
Instruction Fetch

- The CPU fetches instructions from the instruction memory and executes them
- The program counter PC stores the address of the current instruction
- We also need an adder to increase PC
    - MIPS instructions are one word long, so the PC should be incremented by four to read the subsequent instruction



R-type recap

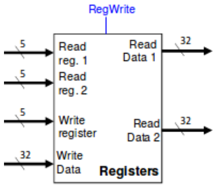
- Arithmetic instructions use R-format
- Three-register-addresss instructions for data manipulations:
  - R:rd← funct rt



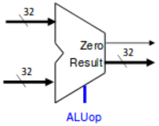
Type	31		format(bits)			0
R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)

Register File and ALU

- R-type instructions require registers and ALU
- There are 32 registers in the MIPS register file.
- We need 5 bits to address registers
- We have 5 operations to encode so ALUop needs to have 3 bits

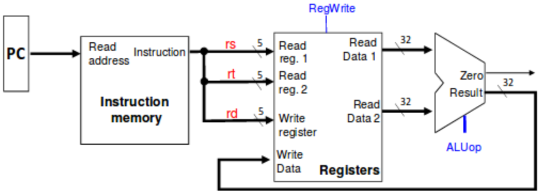


ALUop	Function
000	and
001	or
010	add
110	sub
111	slt



Executing R-type Intructions

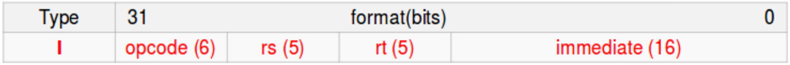
- Instruction is fetched from memory
- rs and rt registers are read from the register file
- The ALU operates on the data read. ALU function determined by the funct field - bits [0:5] of the instruction
- The result is written into the rd



R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)

I-Type Instruction Recap

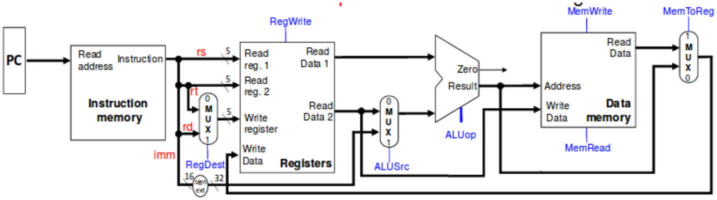
- I-type instructions include lw,sw and beq
- rt is the destination for lw but a source for both sw & beq
  - lw $rt, imm($rs)
  - sw $rt, imm($rs)
  - beq $rs, $rt, imm
- immediate is a 16-bit signed constant (two's complement)



Type	31		format(bits)	0
I	opcode (6)	rs (5)	rt (5)	immediate (16)

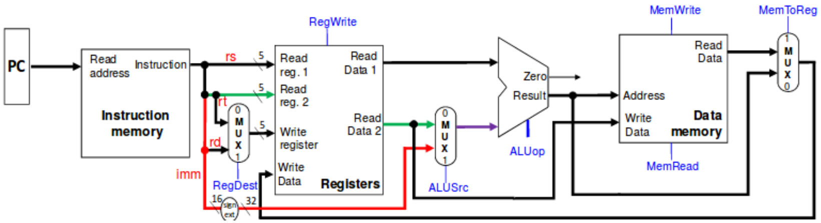
Executing Load/Store Instructions

- For load/store instructions, the base register rs is added to the sign-extended immediate operand producing a data memory address
- The ALU must accept either a register operand for arithmetic instructions or a sign-extended immediate operand for load/store instructions
- We had to add additional three multiplexors with their control signals.



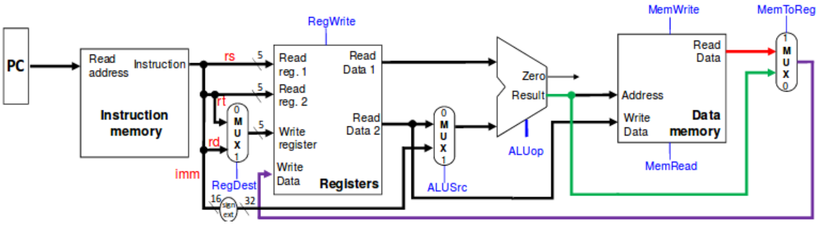
ALUSrc Control Signal

- When asserted selects an immediate operand, otherwise a register operand.



MemToReg Control Signal

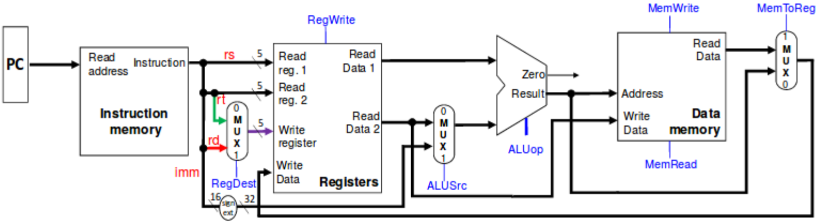
- When asserted selects storing the data data memory contents in a register, otherwise it stores the result of the ALU operation



RegDest Control Signal

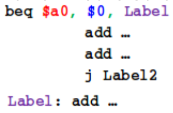
- We need one more mux for selecting the destination register, which is different in R and I-type instructions
- When asserted selects storing the result in a register rd(R-type) otherwise it stores the result in the rt register (lw,I-type).

R	opcode (6)	rs (5)	rt (5)	rd (5)	shamt (5)	funct (6)
I	opcode (6)	rs (5)	rt (5)	immediate (16)		

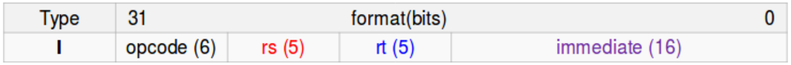


Branch Instruction Recap

- In branch instructions the immediate operand is not an address offset, but the instruction offset:



```
beq $a0, $0, Label
          add …
          add …
          j Label2
Label: add …
```



Type	31		format(bits)	0
I	opcode (6)	rs (5)	rt (5)	immediate (16)

- Label is three instructions after the instruction immediately following beq, hence the offset will be three words (twelve bytes) i.e. the branch address equals PC$+$4$+$imm$\times$4
- Of course, the offset is a signed integer (two's complement)

The End