

Design of the MIPS Processor

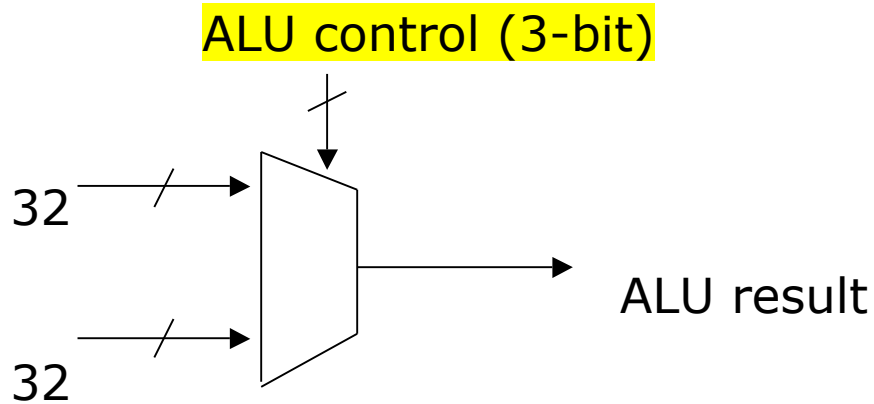
We will study the design of a simple version of MIPS that can support the following instructions:

- I-type instructions LW, SW
- R-type instructions, like ADD, SUB
- Conditional branch instruction BEQ
- J-type branch instruction J

The instruction formats

	6-bit	5-bit	5-bit	5-bit	5-bit	5-bit
LW	op	rs	rt	immediate		
SW	op	rs	rt	immediate		
ADD	op	rs	rt	rd	0	func
SUB	op	rs	rt	rd	0	func
BEQ	op	rs	rt	immediate		
J	op		address			

ALU control

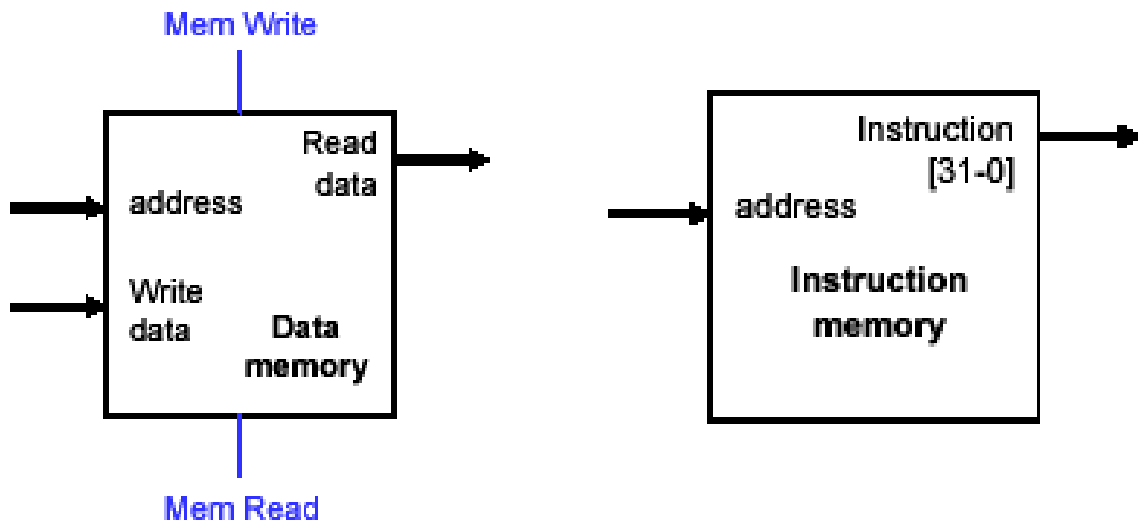


ALU control input	ALU function
000	AND
001	OR
010	add
110	sub
111	Set less than

How to generate the ALU control input? The control unit first generates this from the **opcode** of the instruction.

A single-cycle MIPS

We consider a simple version of MIPS that uses **Harvard architecture**. **Harvard architecture** uses separate memory for instruction and data.



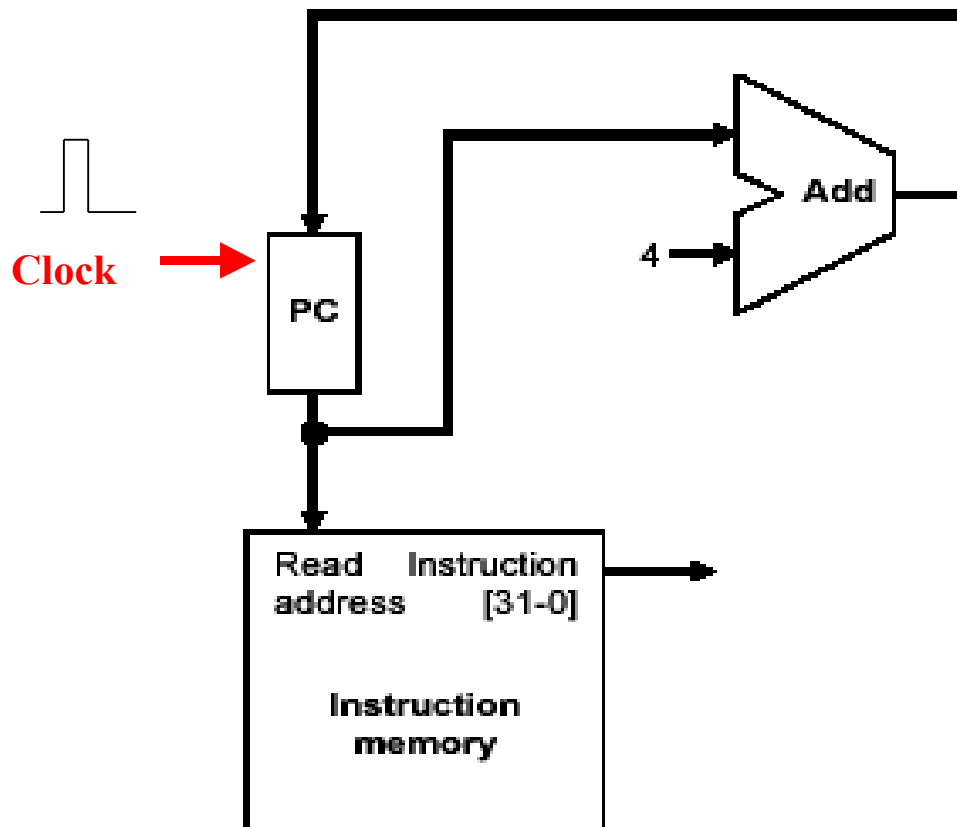
Instruction memory is **read-only** – a programmer cannot write into the instruction memory.

To read from the data memory, set **Memory read** = 1

To write into the data memory, set **Memory write** = 1

Instruction fetching

Each clock fetches I



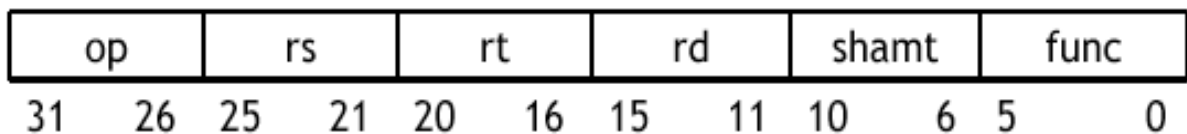
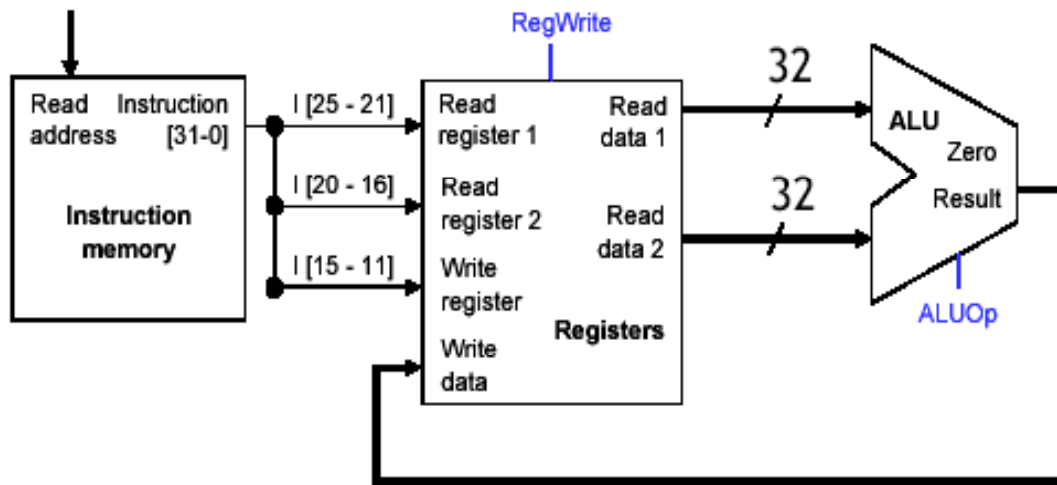
Each clock cycle fetches the instruction from the address specified by the PC, and increments PC by 4 at the same time.

Executing R-type instructions

op	rs	rt	rd	shamt	func
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

add \$s4, \$t1, \$t2 000000 01001 01010 10100 00000 1000000

This is the instruction format for the R-type instructions.



Here are the steps in the execution of an R-type instruction:

Read instruction

Read source registers **rs** and **rt**

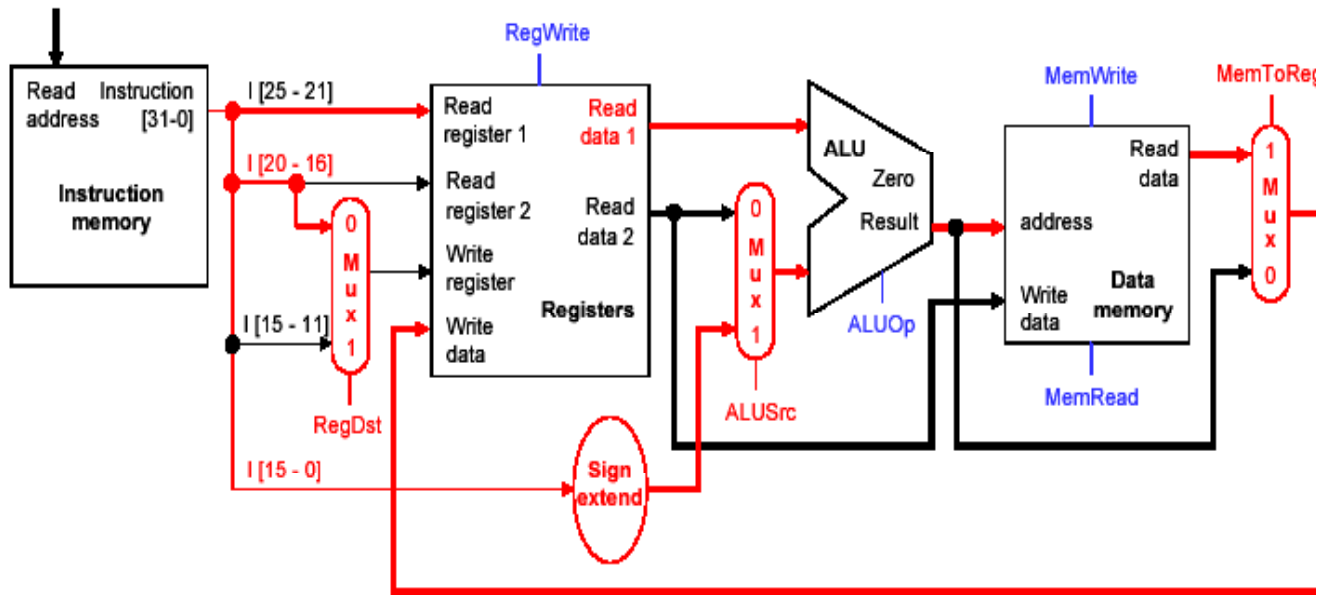
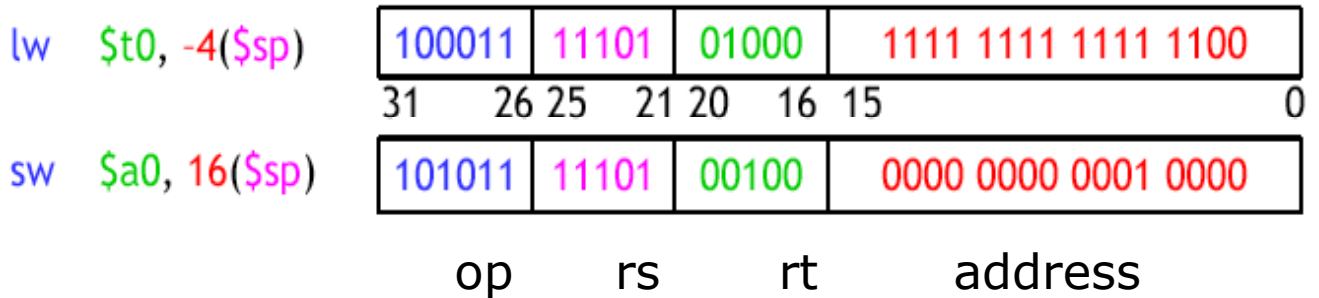
ALU performs the desired operation

Store result in the destination register **rd**.

Q. Why should all these be completed in a single cycle?

Executing *lw*, *sw* instructions

These are I-type instructions.



Try to recognize the steps in the execution of *lw* and *sw*.