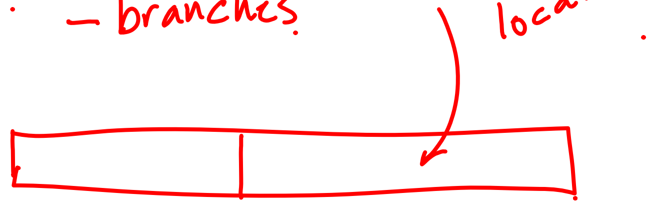
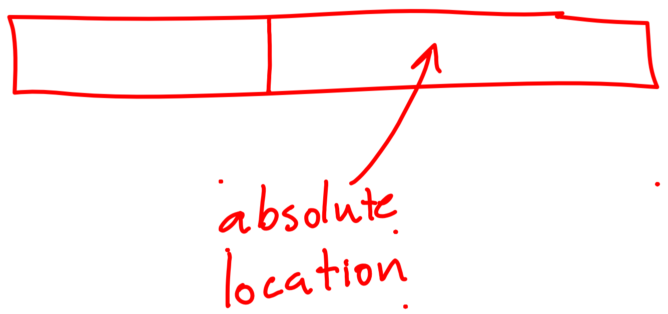
- branches



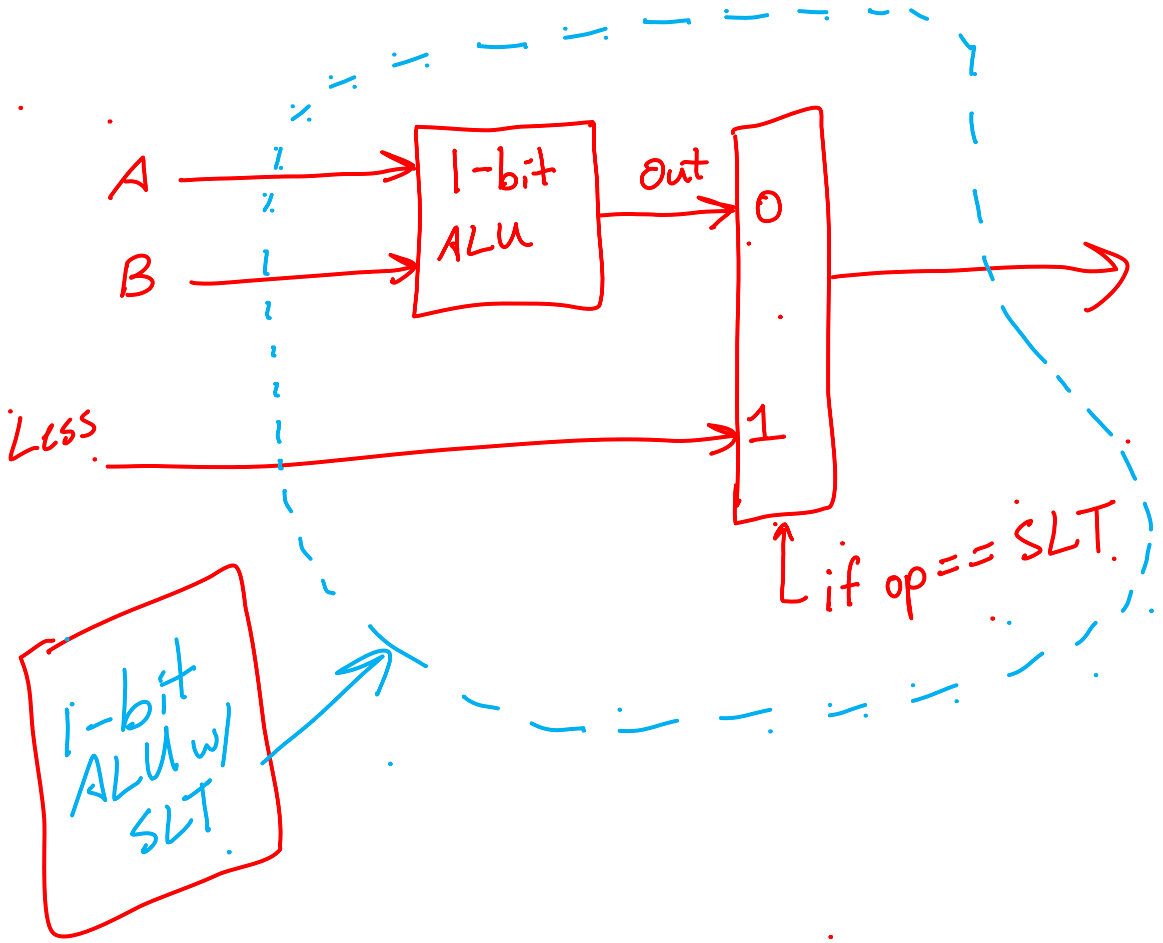
- jumps



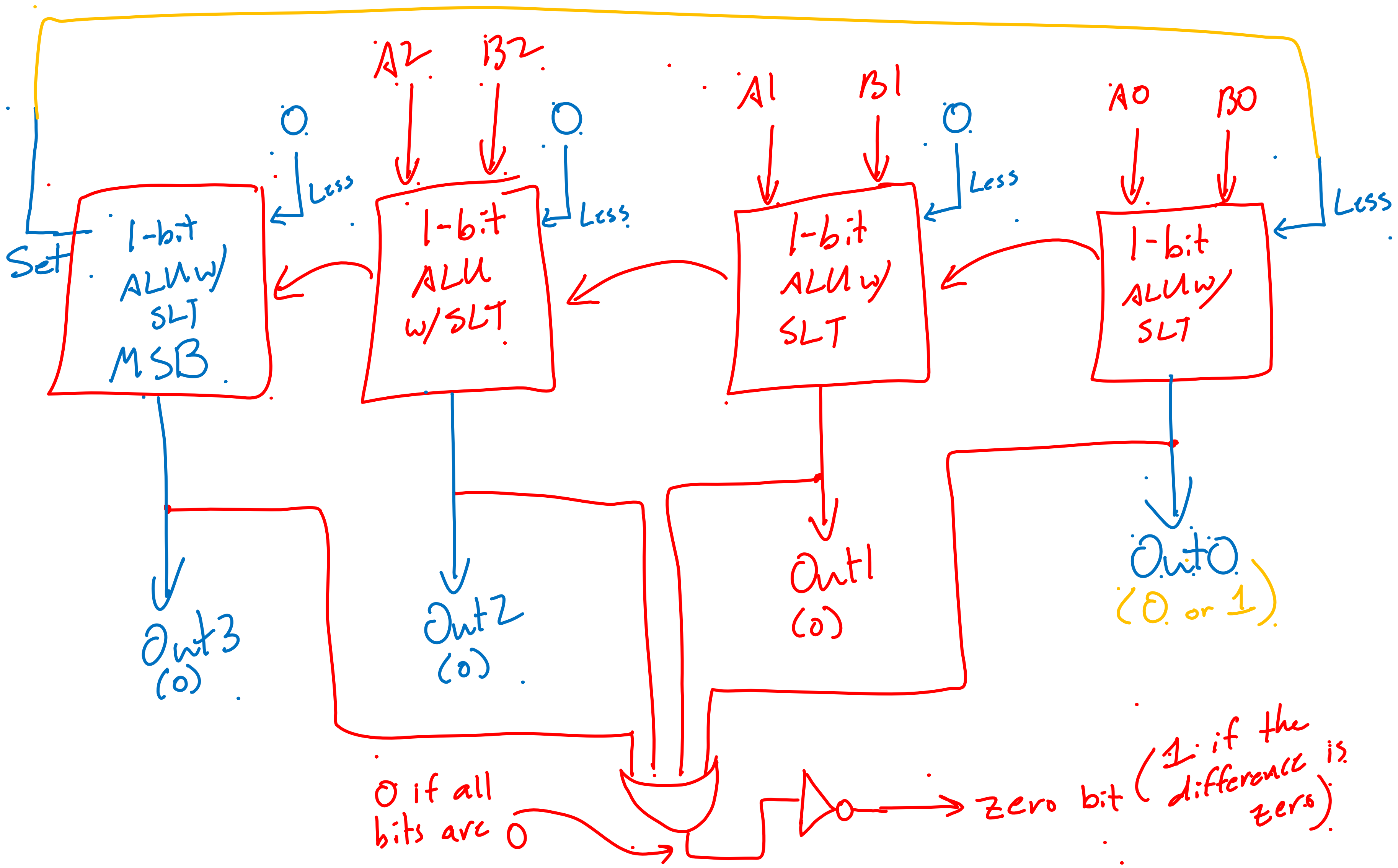
Hardware for Set Less Than

SLT: if (A < B) then Out=1, Out=0

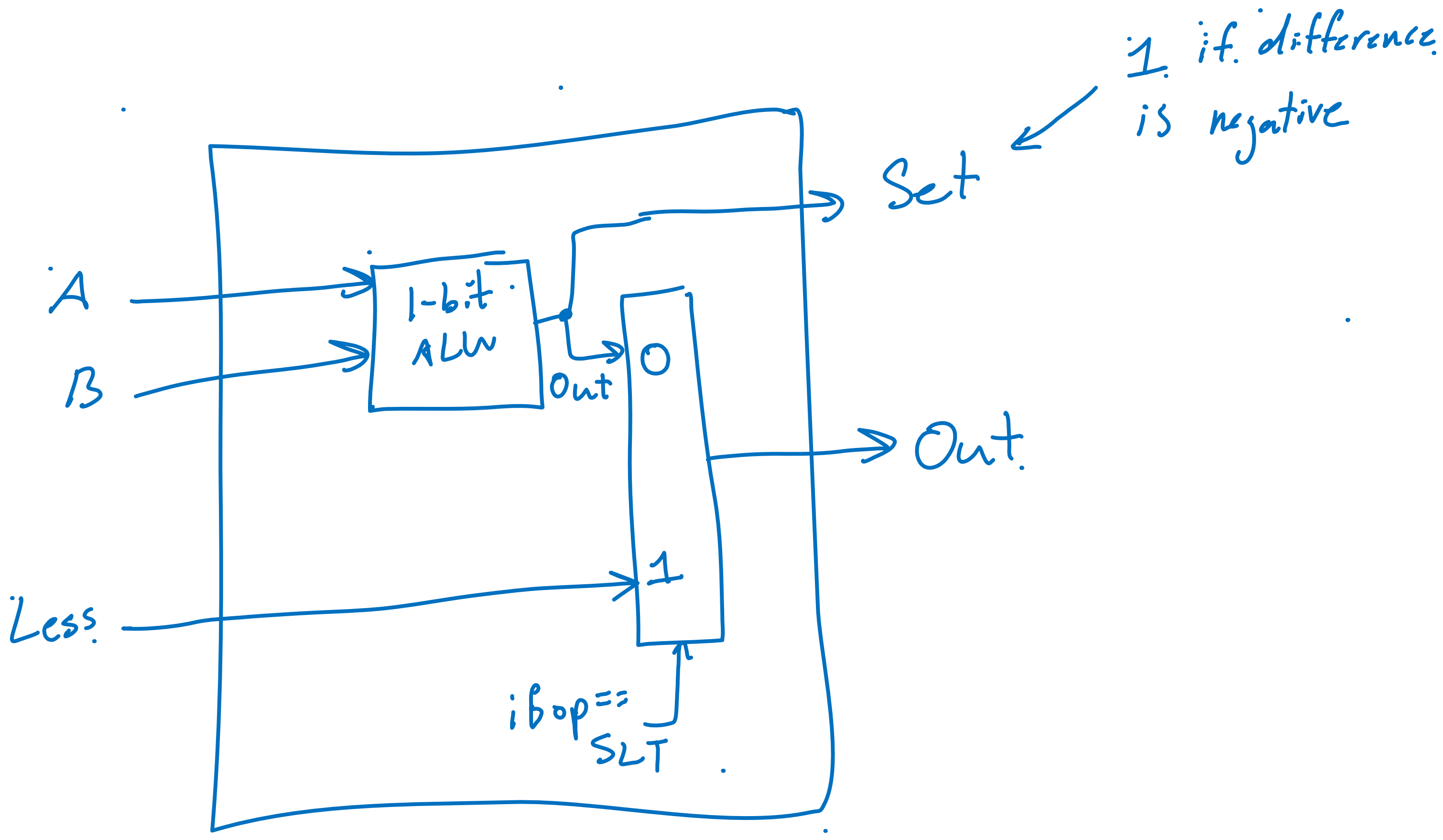
1. on SLT, subtract A-B
2. if ans. is negative, set output to 1



Hardware for Set Less Than



Hardware for Most Significant Bit



Hardware for Set Less Than

Key Points:

1. have the internal ALU's do a subtract
2. take sign bit and use that as the LSB

Hardware for BNE and BEQ

Key Points

1. subtract the register values

2. check zero bit

Hardware for BNE and BEQ

Datapath and Control

~~5.1-5.3~~

4.1-4.4

Goal

- Build an architecture to support the following instructions:
  - add, addi, slt
  - lw, sw
  - j, beq

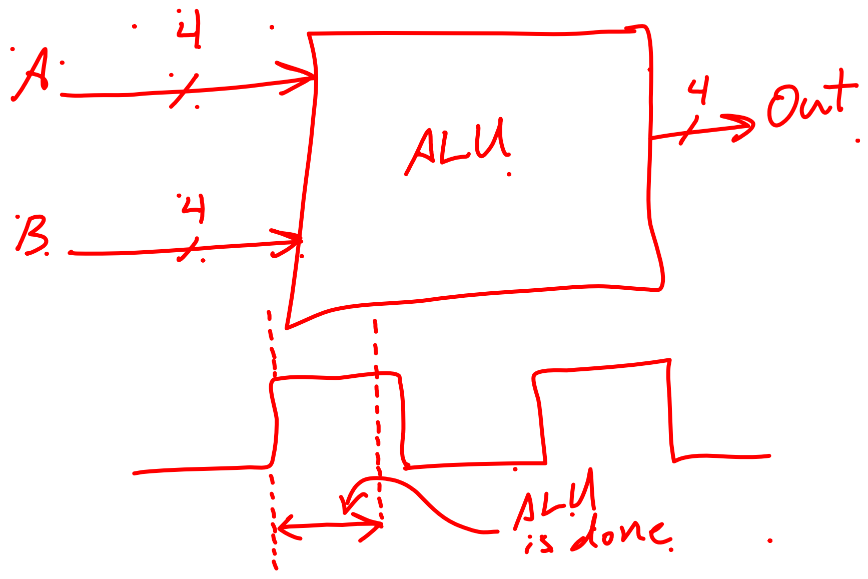
Process

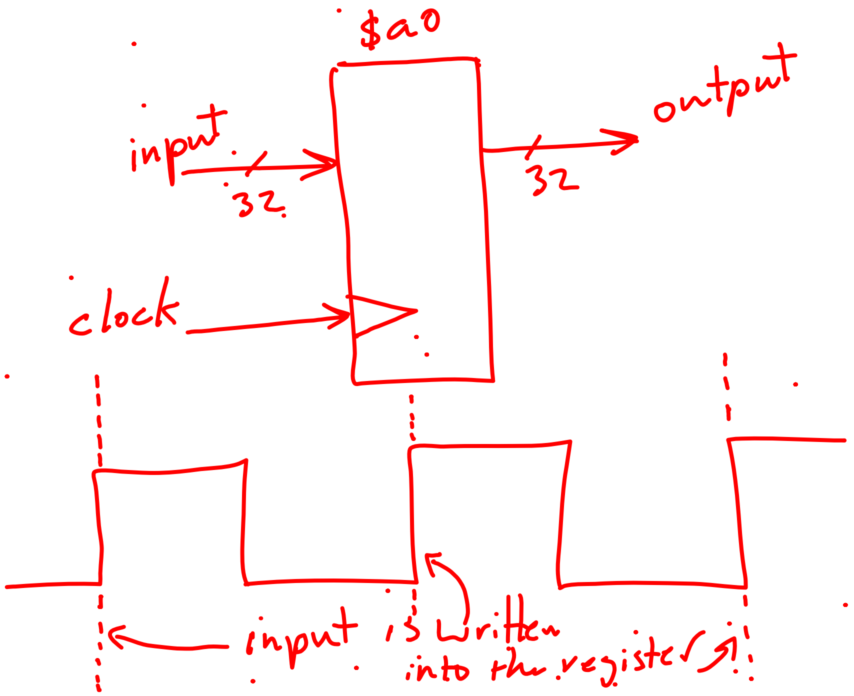
MIPS Instruction Trends

1. store where program where is (program counter)

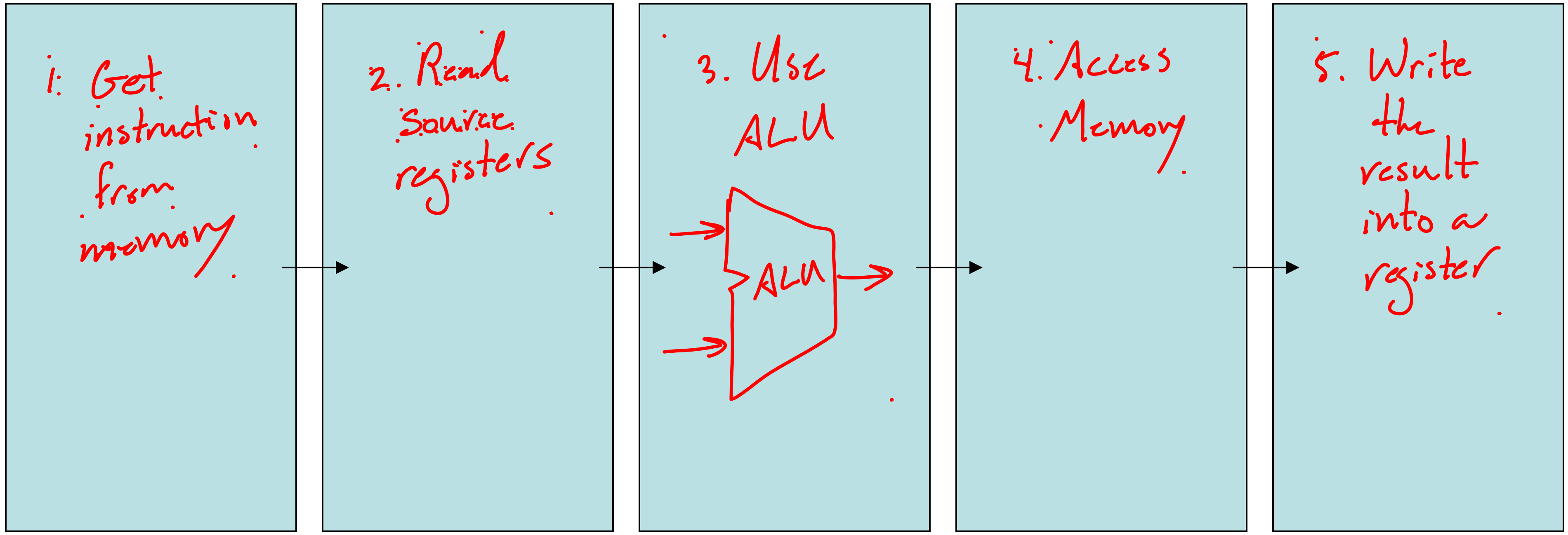
2. some instructions read 1 or 2 registers

3. some instructions write 1 registers

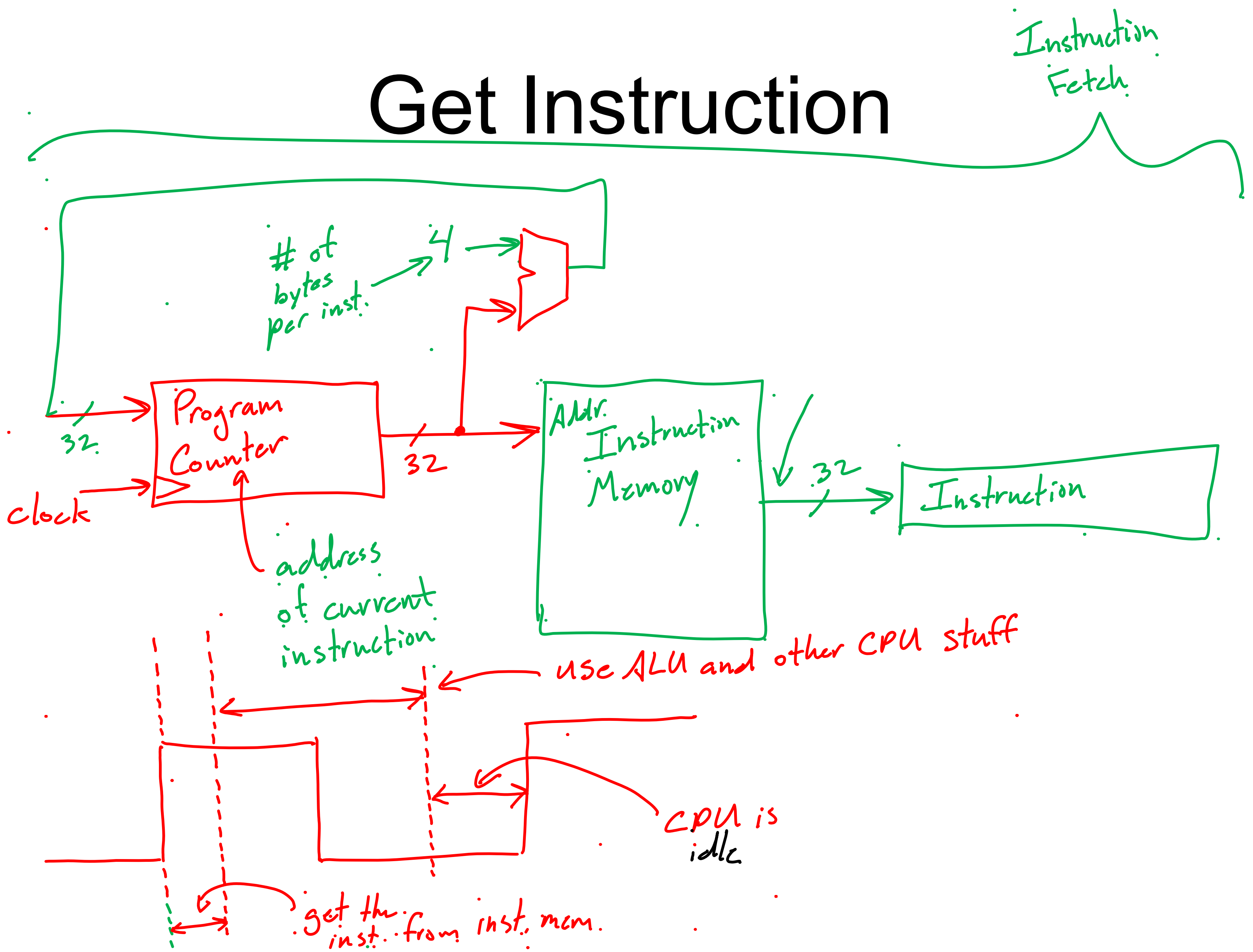




Framework


1: Get instruction from memory
2. Read source registers
3. Use ALU
ALU
4. Access Memory
5. Write the result into a register

Get Instruction



1. Instruction Fetch

- Midterm next Friday
- HW due today beginning of lab

- Key Points:

1. PC is a register (clocked)
2. instruction is read from I.M. and the I.M. is indexed by the PC
3. add 4 to the PC each cycle

- Lab due tonight
  - Makefile required

diff -w -B

- make lab2
- Java: java lab2 file.asm
- C/C++: ./lab2 file.asm

2. Register File

↙

32 registers × 32 bits

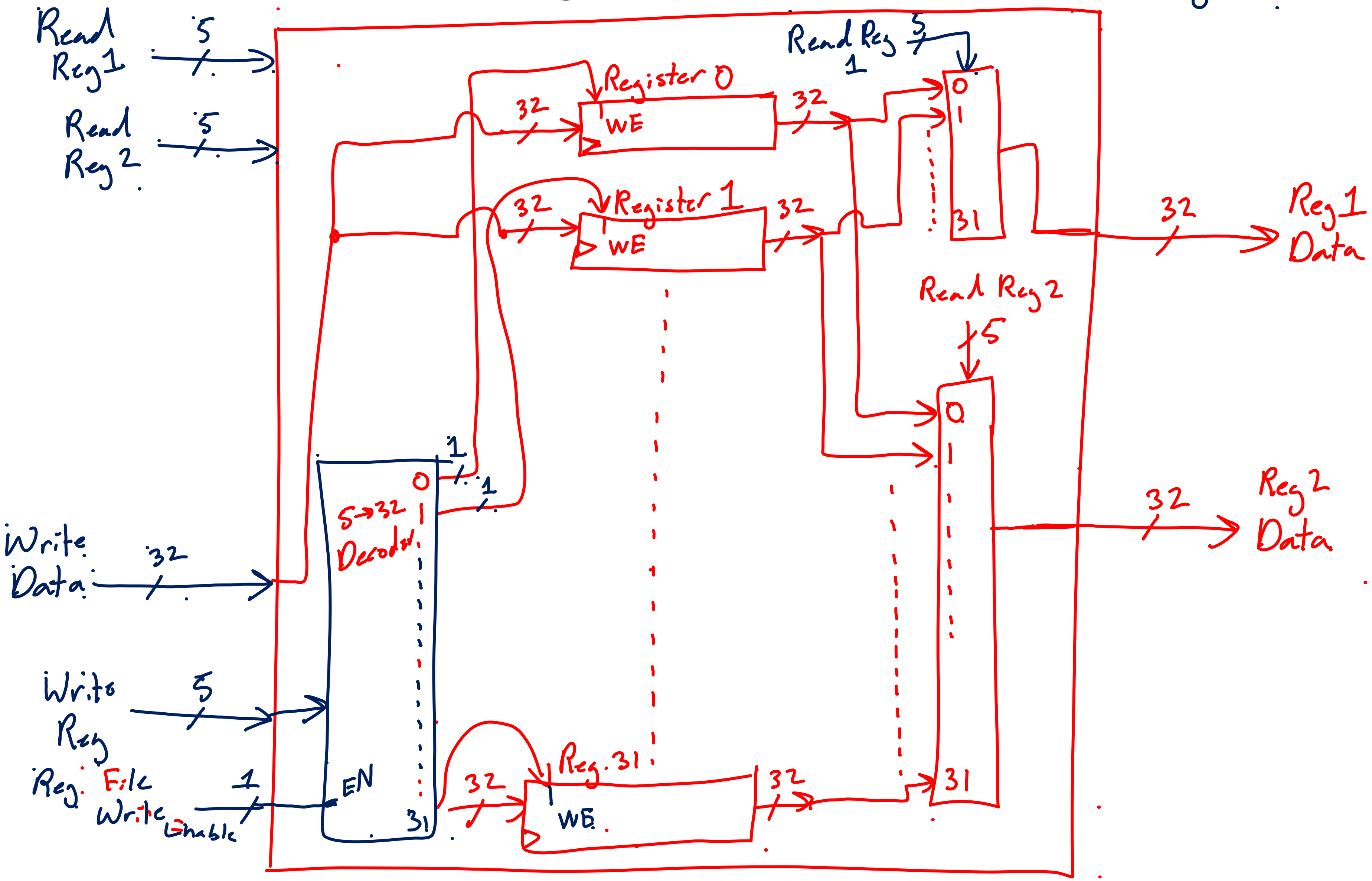
($a0,
$a1, ...)

- Read 2 registers

- Write 1 registers

2. Register File

WE = if 1, then turn on writing for this register


Read Reg 1
5
Read Reg 2
5
Write Data
32
Write Reg
5
Reg. File Write Enable
1
Register 0
WE
32
Register 1
WE
32
Reg. 31
WE
32
5→32 Decoder
0
1
31
EN
1
1
Read Reg 1
5
0
1
31
Read Reg 2
5
0
1
31
32
Reg 1 Data
32
Reg 2 Data