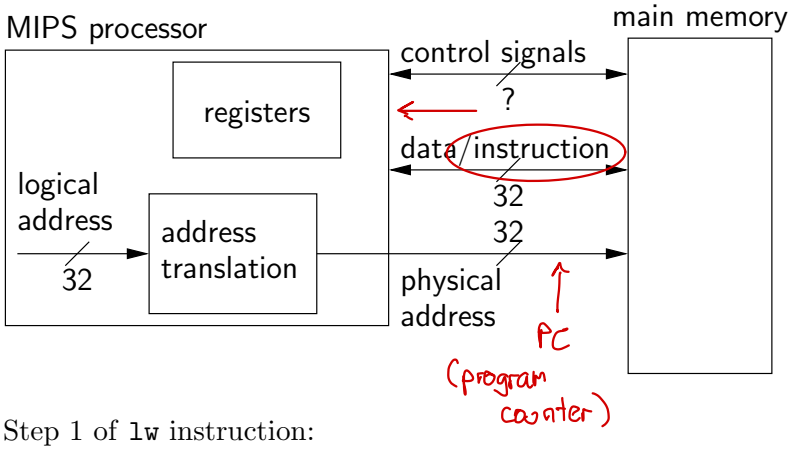
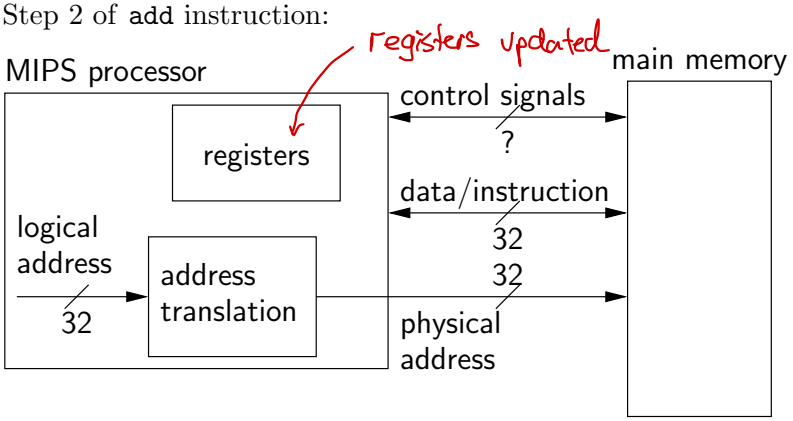
ENCM 369 Winter 2020: Handout for lectures Mon Jan 20

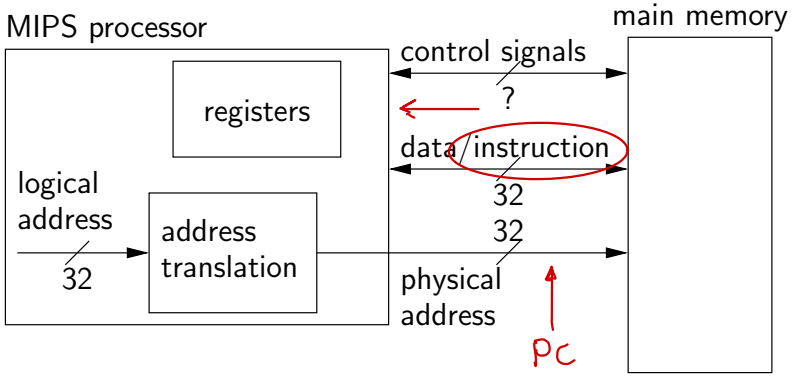
Step 1 of `add` instruction:



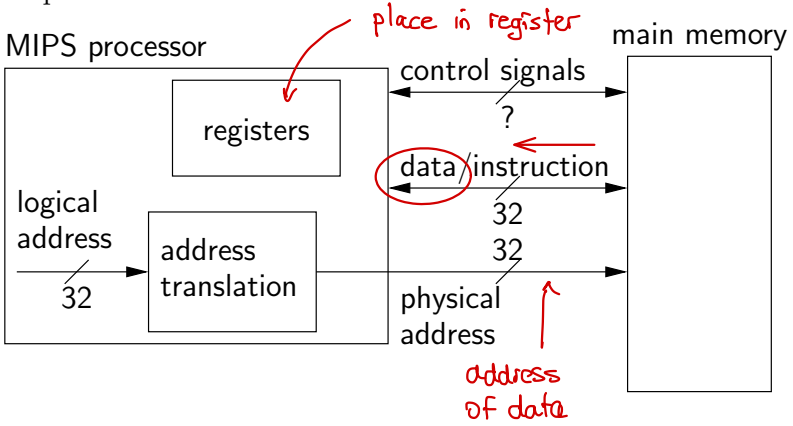
Step 2 of `add` instruction:



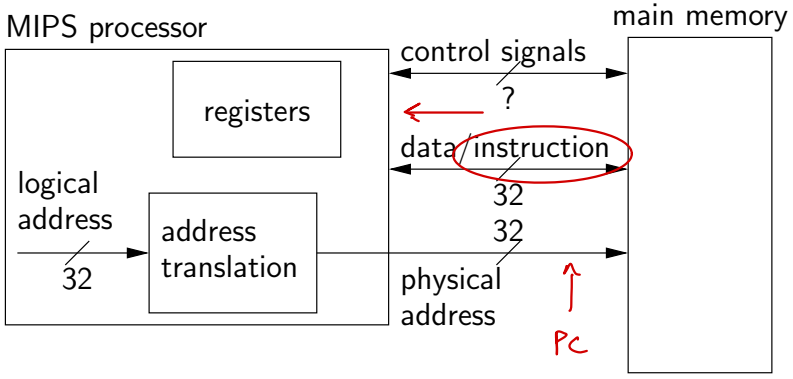
Step 1 of `lw` instruction:



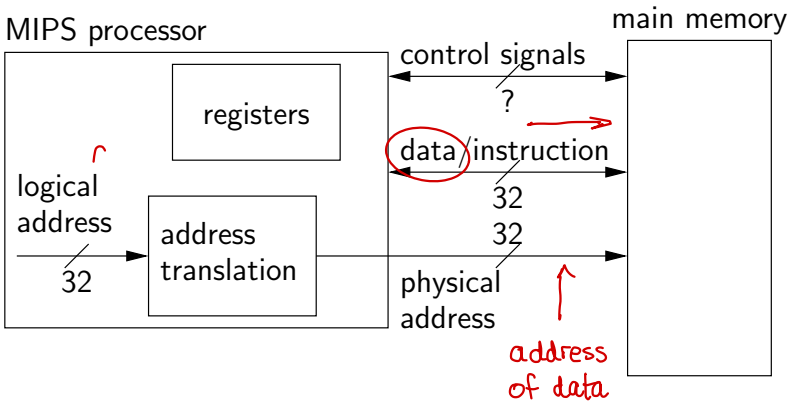
Step 2 of `lw` instruction:



Step 1 of `sw` instruction:



Step 2 of `sw` instruction:



More sample instructions

```
addi   $10, $18, 1025
```

result in GPR 10

add constant 1025 to GPR 18

```
001000 - 10010 - 01010 - 0000_0100_0000_0001
```

code to add GPR and a constant; place result in a GPR

GPR 18

GPR 10

1025_{10} in binary (16-bit 2's-complement representation)

"ell"

```
lw   $8, (4)
```

load word

GPR 8

GPR 4 contains the address of the data to fetch

```
100011 - 00100 - 01000 - 0000_0000_0000_0000
```

Code to copy word from memory to a register

GPR 4

GPR 8

offset added to GPR 4, and result is used as the pointer

- more about this later!

Processor/memory interaction for some sample instructions

Today's handout

Why assembly language?

- Hand-written code tended to be smaller and faster than compiler-generated code.
- less so now. Optimizing compilers now very good at it!

So why now?

1. Highly-optimized pieces of hand-written code can still be more efficient.

2. Understand others' old code
3. Principal reason for ENCM 369:

   - understand what a processor does
   - seeing what a compiler does

Disadvantages to assembly language programming

1. Cost — slower to write code, hard to read debug
2. Portability — it is processor specific.

Register names and uses

$0 — The zero register; contents are always zero. Called $zero.

```
add   $6, $zero, $zero
```

sets GPR6 to zero

```
add   $zero, $6, $7
```

not so useful as a destination register — always zero.

$16 – $23

Known as $s0 – $s7 ⟶ generally used as local variables in procedures (i.e. C functions)

- optimizing compilers will use these if they can for local variables; otherwise, variables go on the stack.

$8 – $15, $24, $25

Known as $t0 – $t9 ⟶ used as temporary variables (intermediate results)

Examples of local variables and temporaries

Suppose we have a C function with

```
int a, b, c, d, e, f;
```

Suppose these are allocated as follows:

var	reg
a	$s0
b	$s1
c	$s2
d	$s3
e	$s4
f	$s5

C-code	translation
a = 0;	add $s0, $zero, $zero

b = (c-d) + (e-f)	sub $t0, $s2, $s3 # (c-d)
	sub $t1, $s4, $s5 # (e-f)
	add $s1, $t0, $t1

introduces a comment