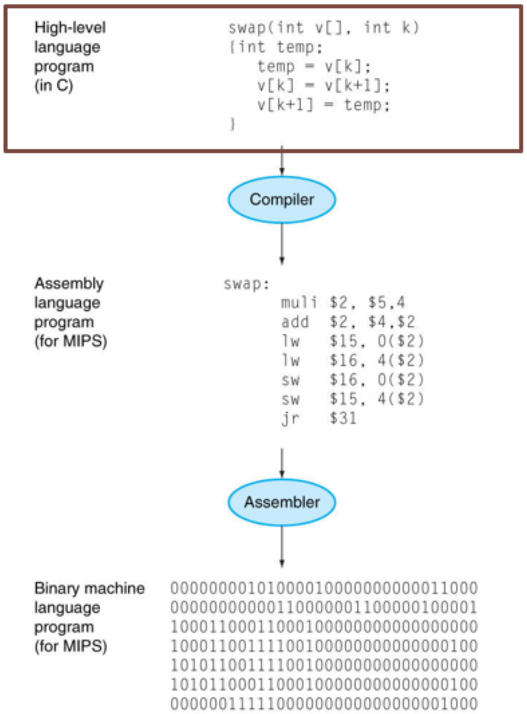
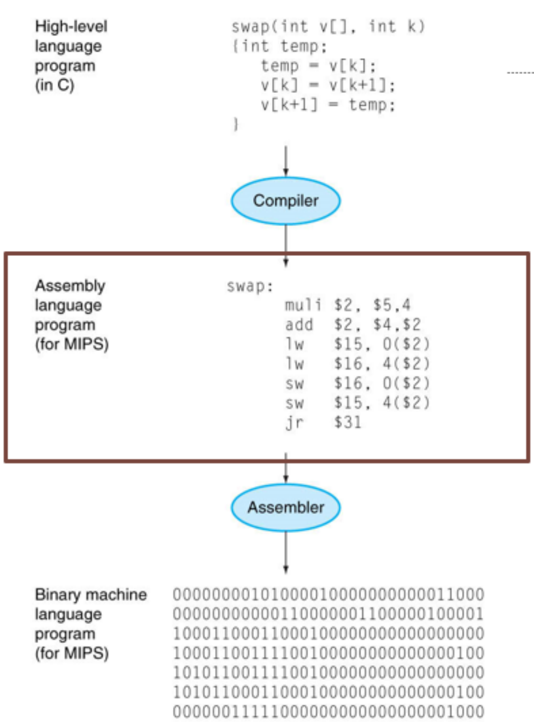
7 - The Processor: Datapath

7.1 Brief Recap



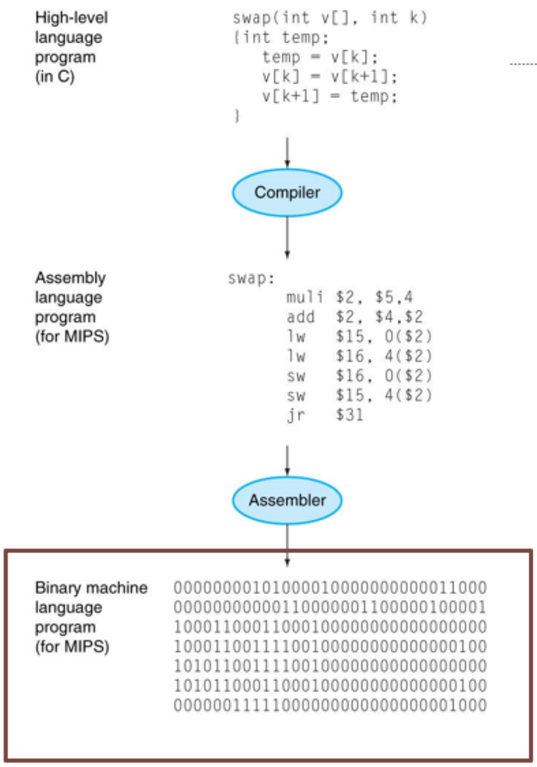
- Write program in high-level language (e.g., C)

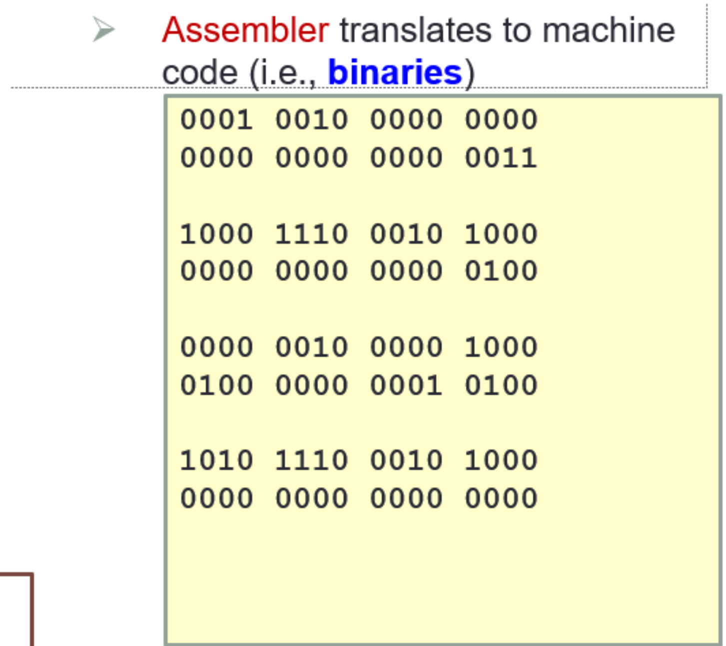
```
if(x != 0) {
  a[0] = a[1] + x;
}
```

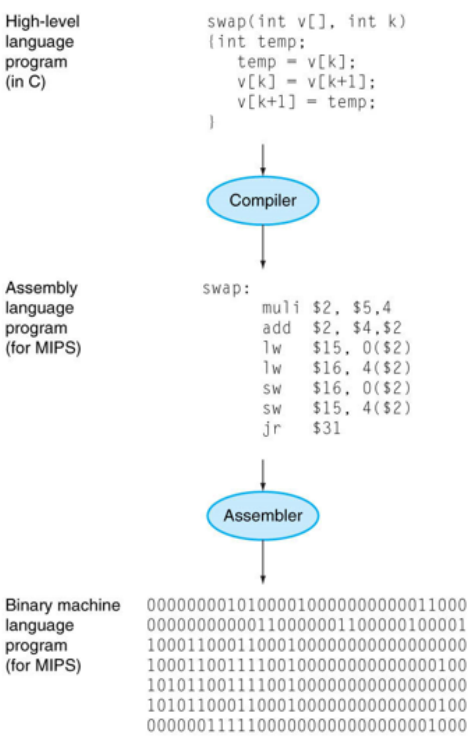


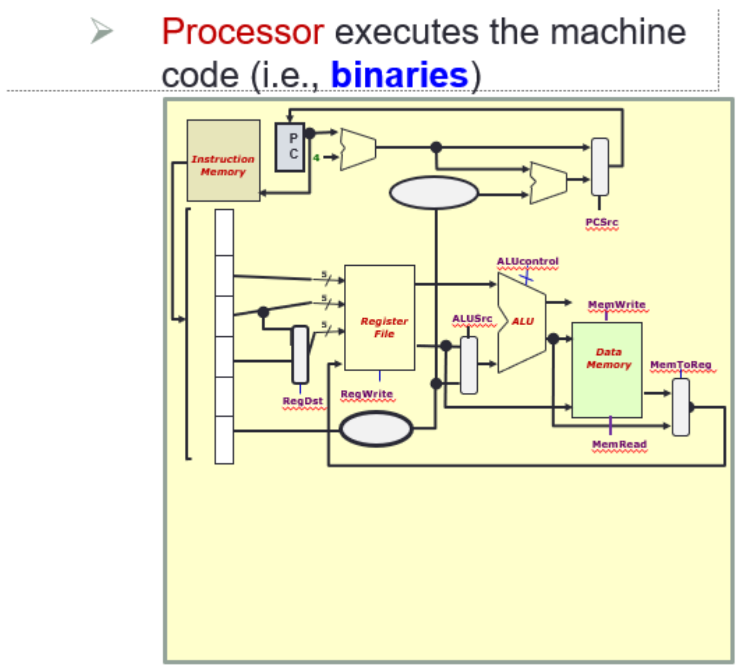
- Compiler translates to assembly language (e.g., MIPS)

```
      beq $16, $0, Else
      lw  $8, 4($17)
      add $8, $8, $16
      sw  $8, 0($17)
Else:
```









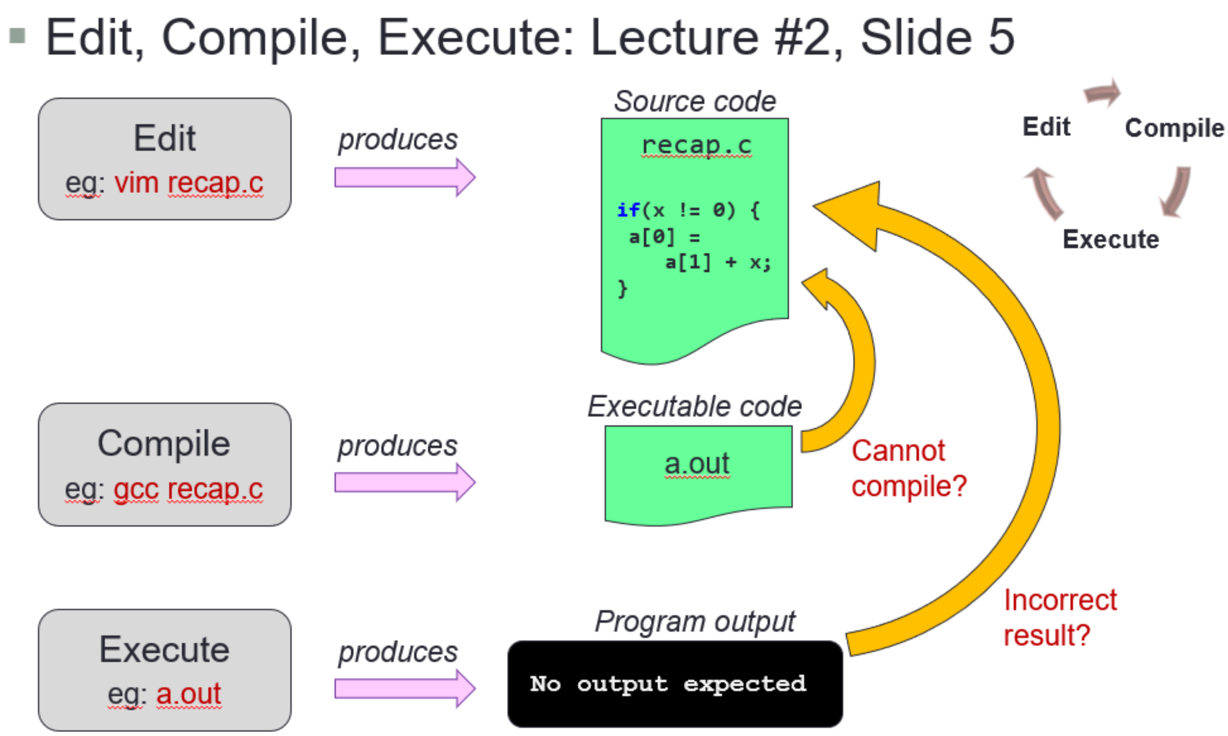
7.2 From C to Execution

- We play the role of Programmer, Compiler, Assembler, and Processor
    - Program:

```c
if (x != 0) {
    a[0] = a[1] + x;
}
```

- Programmer: Show the workflow of compiling, assembling, and executing C program
- Compiler: Show how the program is compiled into MIPS
- Assembler: Show how the MIPS is translated into binaries
- Processor: Show how the datapath is activated in the processor

7.2.1 Writing C Program



7.2.2 Compiling to MIPS

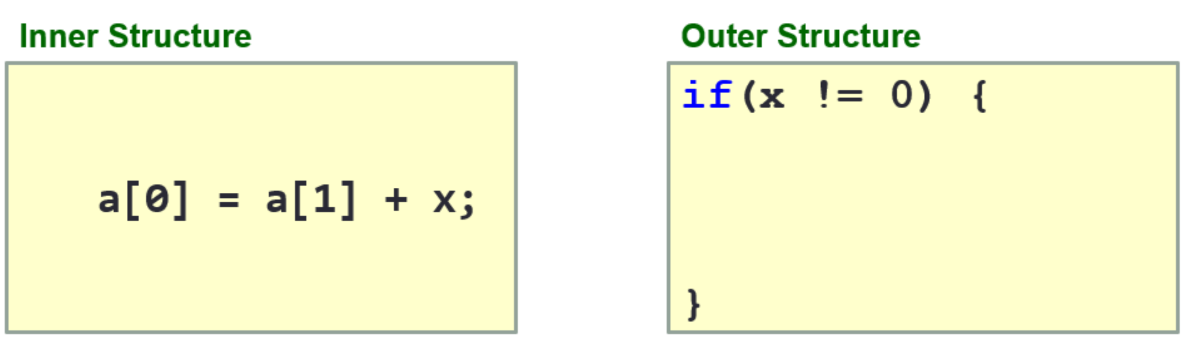
Key Idea

- Key Idea #1:

  Compilation is a structed process

  ```
  if (x != 0) {
      a[0] = a[1] + x;
  }
  ```

  Each structure can be compiled independently



- Key Idea #2:

  Variable-to-Register Mapping

  Let the mapping be:

Variable	Register Name	Register Number
x	$s0	$16
a	$s1	$17

Common Technique

- Common Technique #1:

  Invert the condition for shorter code

Outer Structure

```
if(x != 0) {


}
```

Outer MIPS Code

```
      beq $16, $0, Else

      # Inner Structure

Else:
```

- Common Technique #2:

  Break complex operations, use temp register

Inner Structure

```
a[0] = a[1] + x;
```

Simplified Inner Structure

```
$t1  = a[1];
$t1  = $t1 + x;
a[0] = $t1;
```

- Common Technique #3:

  Array access is `lw`, array update is `sw`

Simplified Inner Structure

```
$t1  = a[1];
$t1  = $t1 + x;
a[0] = $t1;
```

Inner MIPS Code

```
lw  $8, 4($17)
add $8, $8, $16
sw  $8, 0($17)
```

Common Error

- Common Error #1:

  Assume that the address of the next word can be found by incrementing the address in a register by 1 instead of by the word size in bytes

  Example:

  `$t1 = a[1]`

  is translated to:

  `lw $8, 4($17)`

  instead of

  `lw $s8, 1($17)`

Finalize

- Last Step:

  Combine the two structures logically

Inner MIPS Code

```
lw   $8, 4($17)
add  $8, $8, $16
sw   $8, 0($17)
```

Outer MIPS Code

```
      beq $16, $0, Else

      # Inner Structure

Else:
```

Combined MIPS Code

```
      beq $16, $0, Else
      lw   $8, 4($17)
      add  $8, $8, $16
      sw   $8, 0($17)
Else:
```

7.2.3 Assembling to Binaries

- Instruction Types Used:

  1. R-Format: `opcode $rd, $rs, $rt`

6	5	5	5	5	6
opcode	rs	rt	rd	shamt	funct

  2. I-Format: `opcode $rt, $rs, immediate`

6	5	5	16
opcode	rs	rt	immediate

  3. Branch:

     - Use I-format

- PC = (PC+4) + (`immediate` x 4)

4. `beq $16, $0, Else`
    - Compute immediate value
        - `immediate` = 3
    - Fill in fields

6	5	5	16
4	16	0	3

    - Convert to binary

6	5	5	16
4	16	0	3

```
      beq $16, $0, Else
      lw  $8, 4($17)
      add $8, $8, $16     +3
      sw  $8, 0($17)
Else:
```

5. `lw $8, 4($17)`
    - Filled in fields (Refer to MIPS Reference data)

6	5	5	16
35	17	8	4

    - Convert to binary

100011	10001	01000	0000000000000100

```
0001 0010 0000 0000 0000 0000 0000 0011
      lw  $8, 4($17)
      add $8, $8, $16
      sw  $8, 0($17)
Else:
```

6. `add $8, $8, $16`
    - Filled in fields

6	5	5	5	5	6
0	8	16	8	0	32

    - Convert to binary

000000	01000	10000	01000	00000	100000

```
0001 0010 0000 0000 0000 0000 0000 0011
1000 1110 0010 1000 0000 0000 0000 0100
        add $8, $8, $16
        sw  $8, 0($17)
Else:
```

7. `sw $8, 0($17)`
    - Filled in fields

6	5	5	16
43	17	8	0

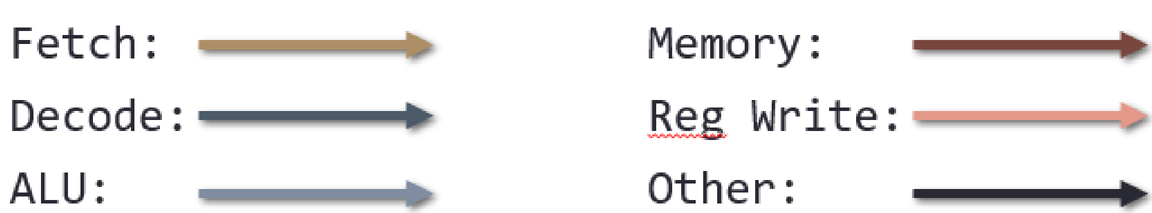
    - Convert to binary

101011	10001	01000	0000000000000000

```
0001 0010 0000 0000 0000 0000 0000 0011
1000 1110 0010 1000 0000 0000 0000 0100
0000 0001 0001 0000 0100 0000 0010 0000
        sw  $8, 0($17)
Else:
```

7.2.4 Execution (Datapath)

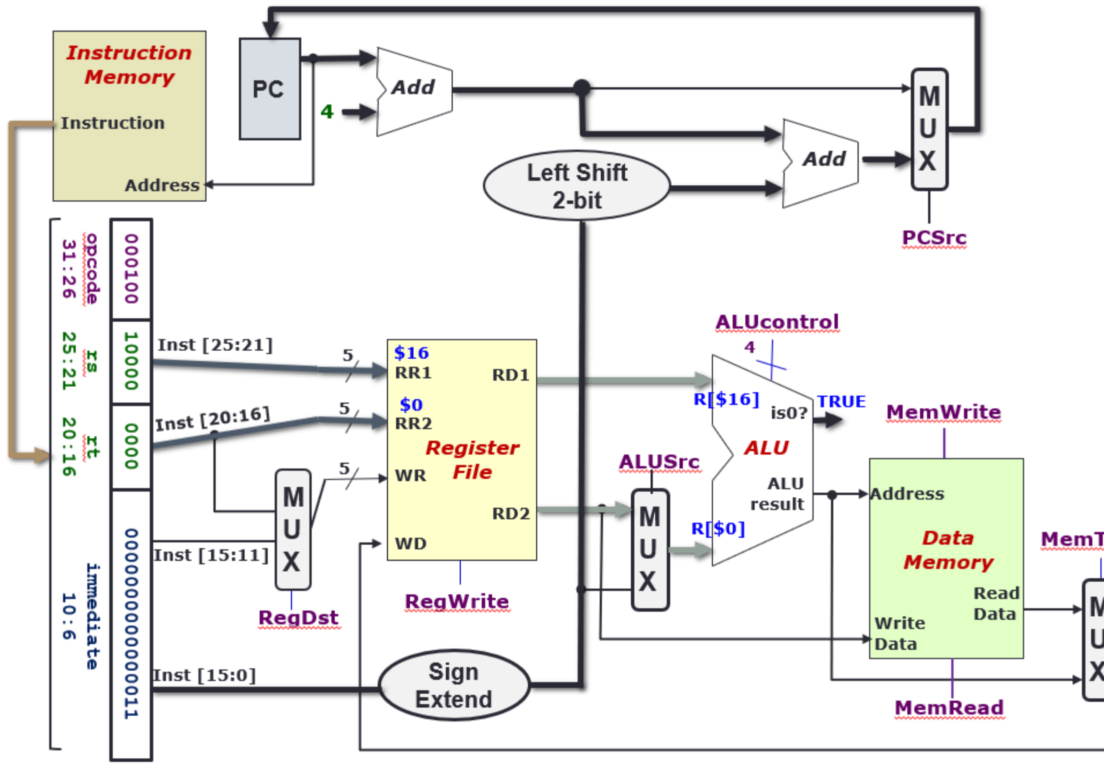
- Given the binary
    - Assume two possible executions
        1. `$16 == $0` (shorter)
        2. `$16 != $0` (larger)
    - Convention:



Lecture #7a: The Processor: Datapath

- Assume $16 == $0

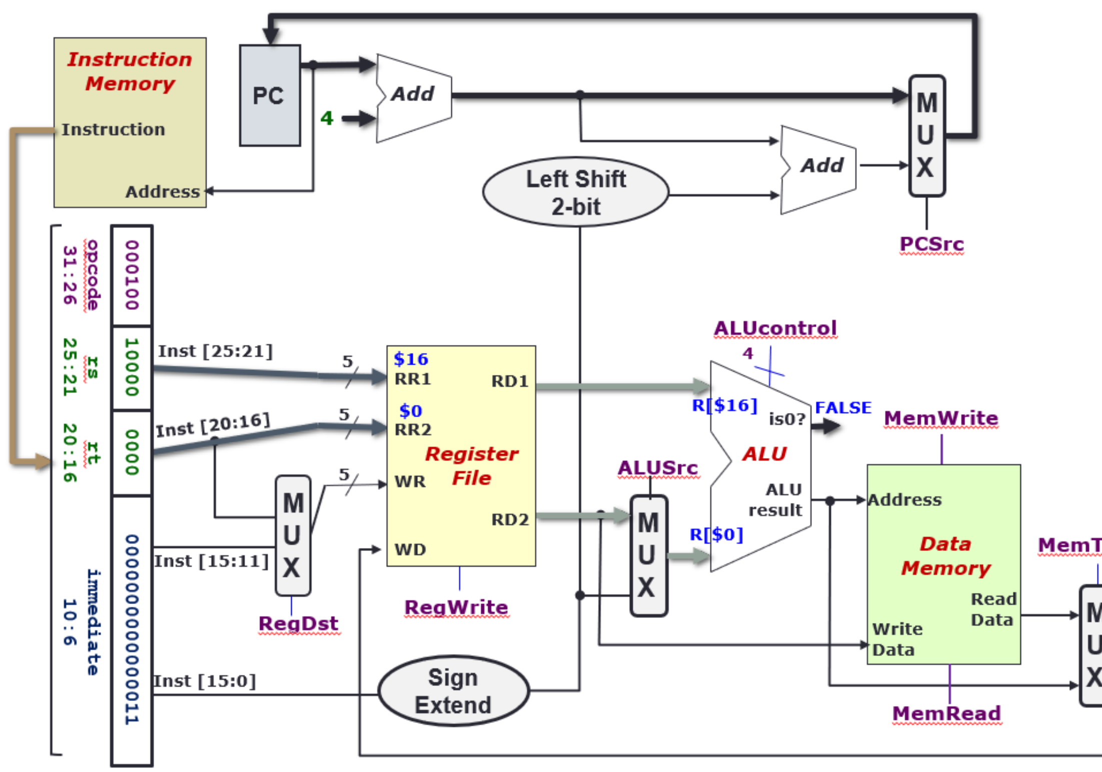
beq $16, $0, Else



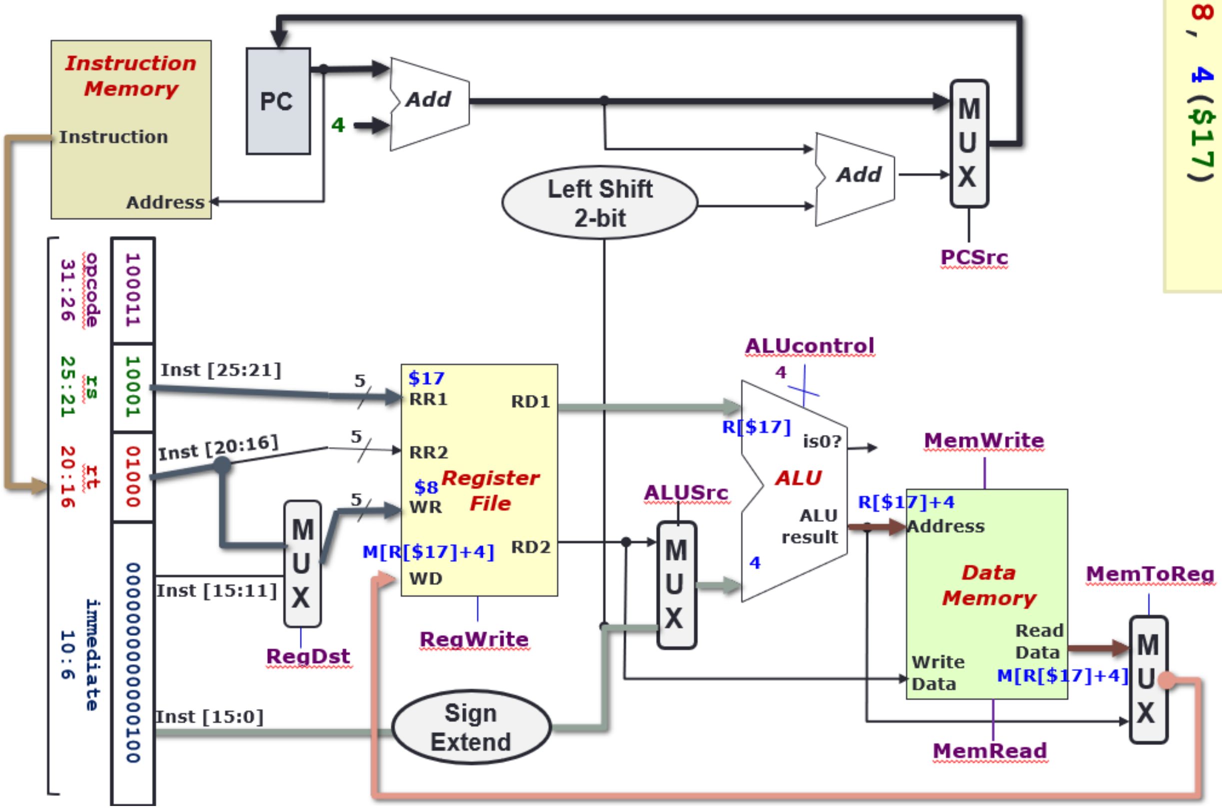
Lecture #7a: The Processor: Datapath

- Assume $16 != $0

beq $16, $0, Else

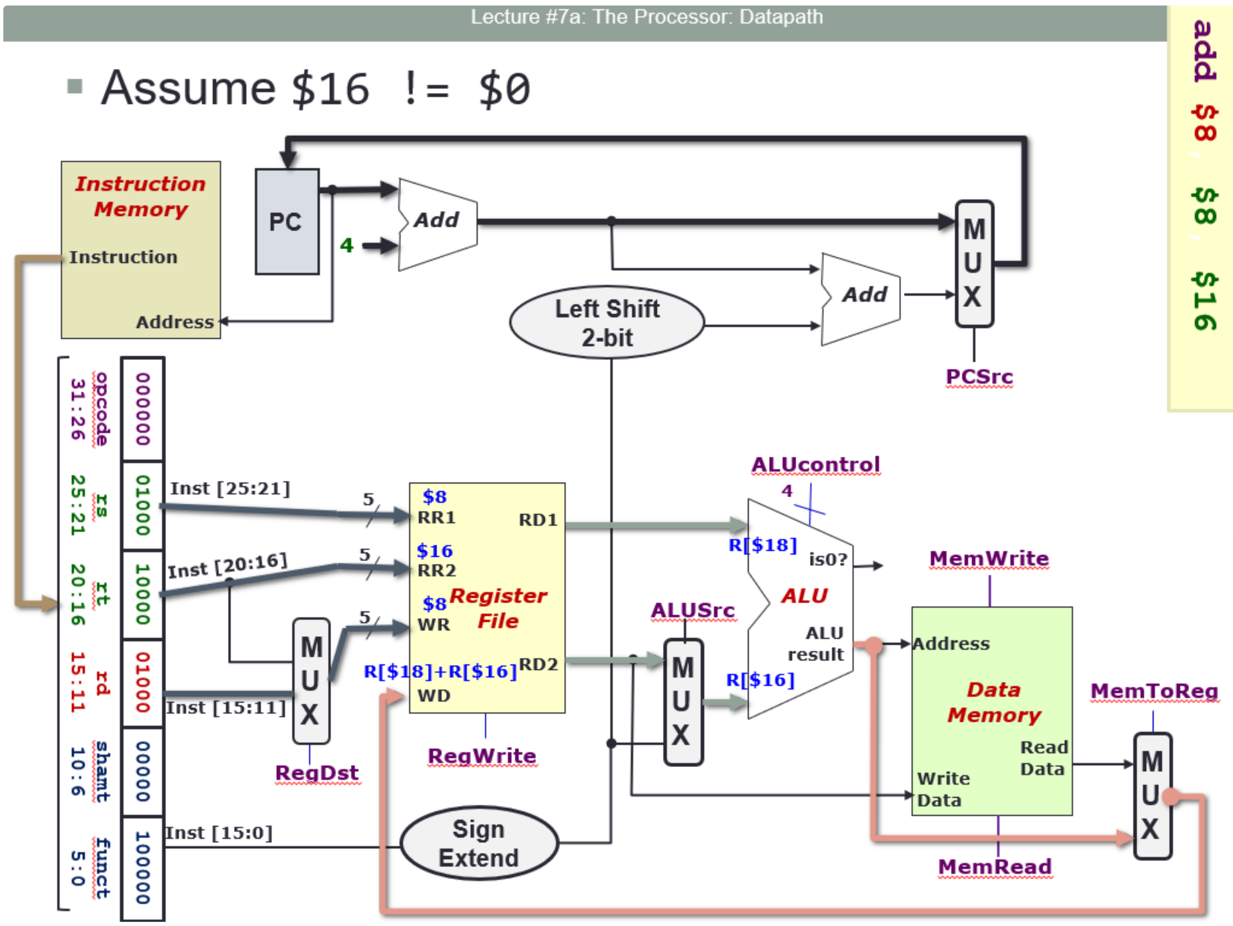


- Assume $16 != $0



Lecture #7a: The Processor: Datapath

- Assume $16 != $0



sw $8, 0($17)

- Assume $16 != $0