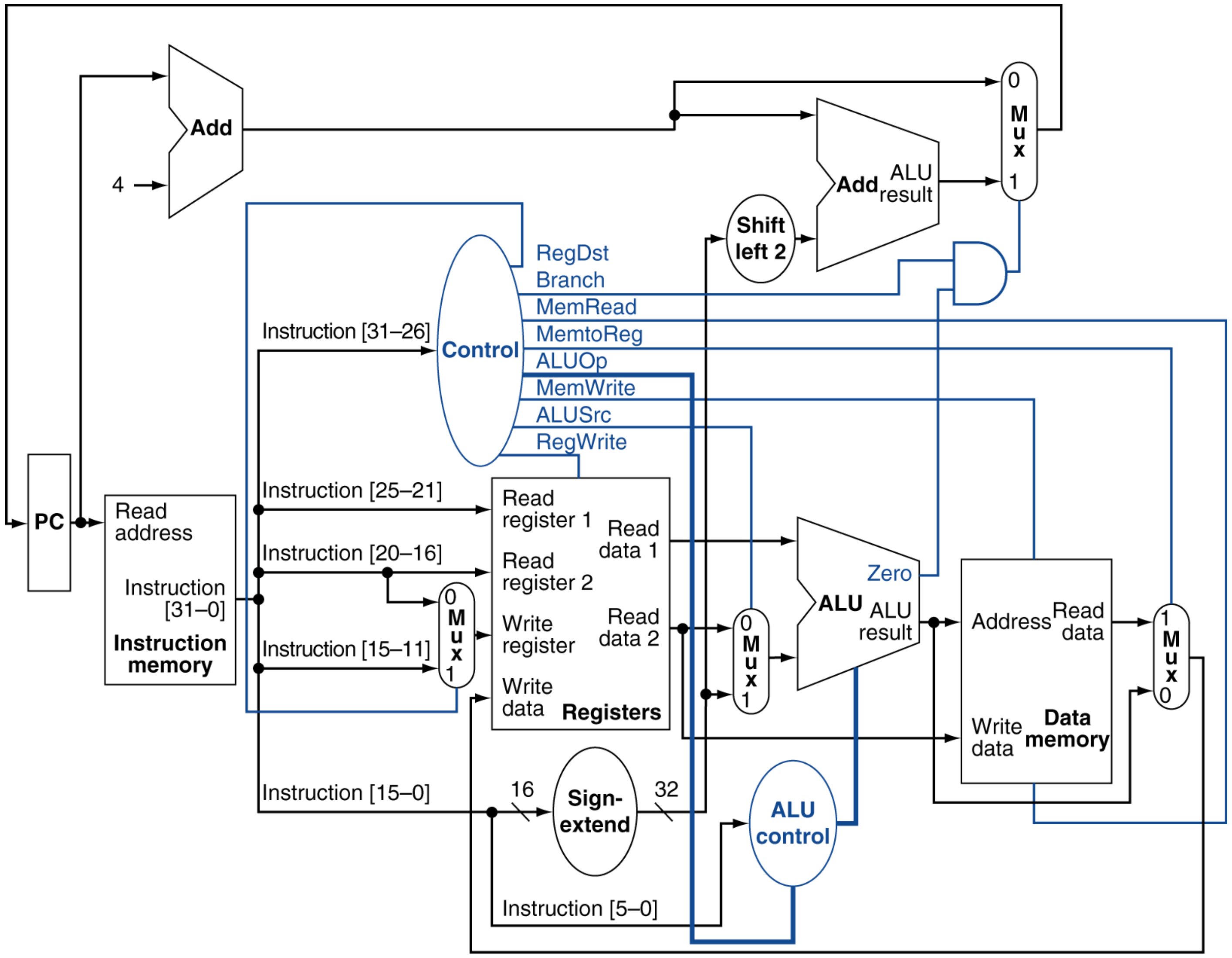
FU05 Computer Architecture

8. Controlpath (制御パス)

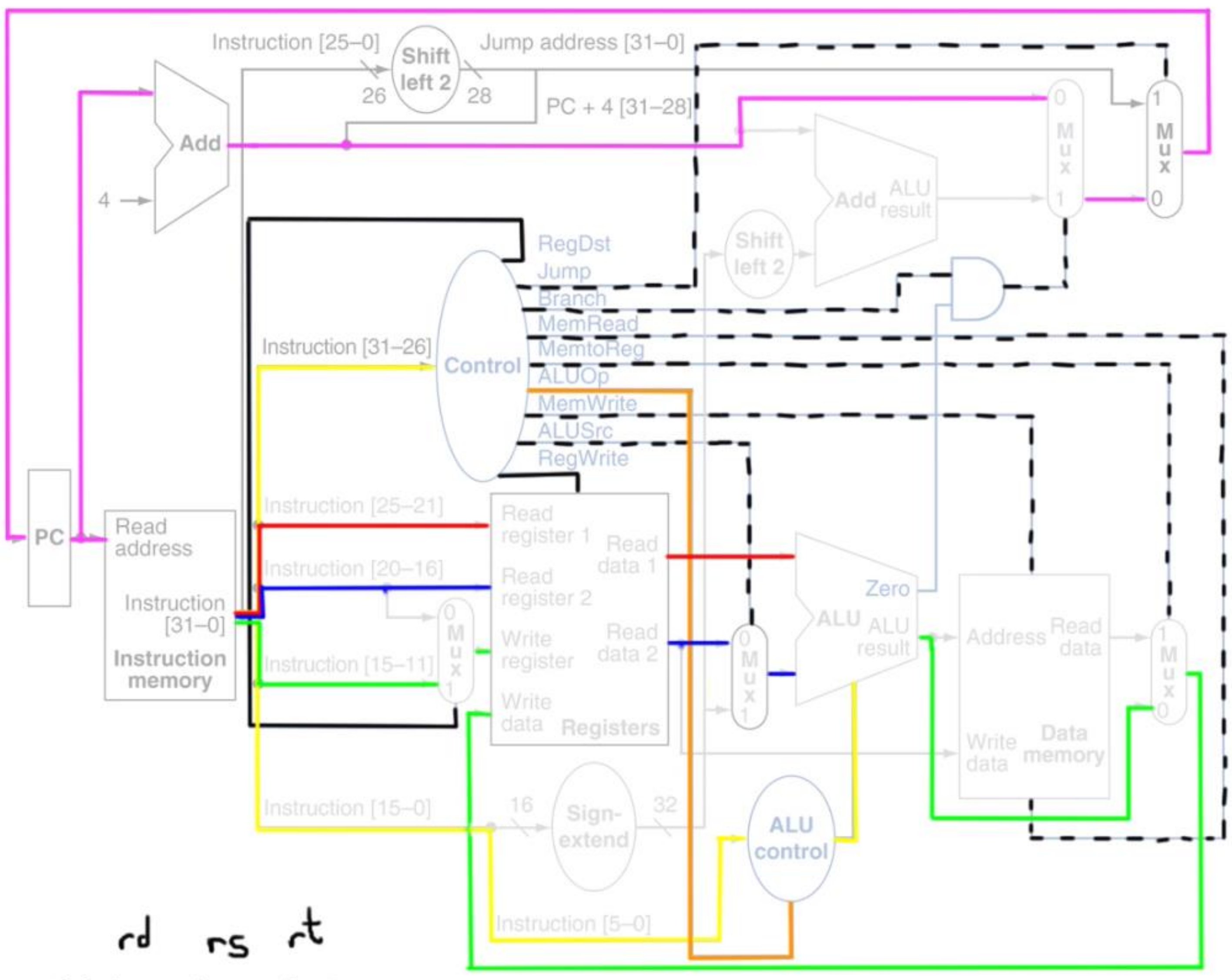
Ben Abdallah Abderazek

E-mail: benab@u-aizu.ac.jp

Datapath With Control



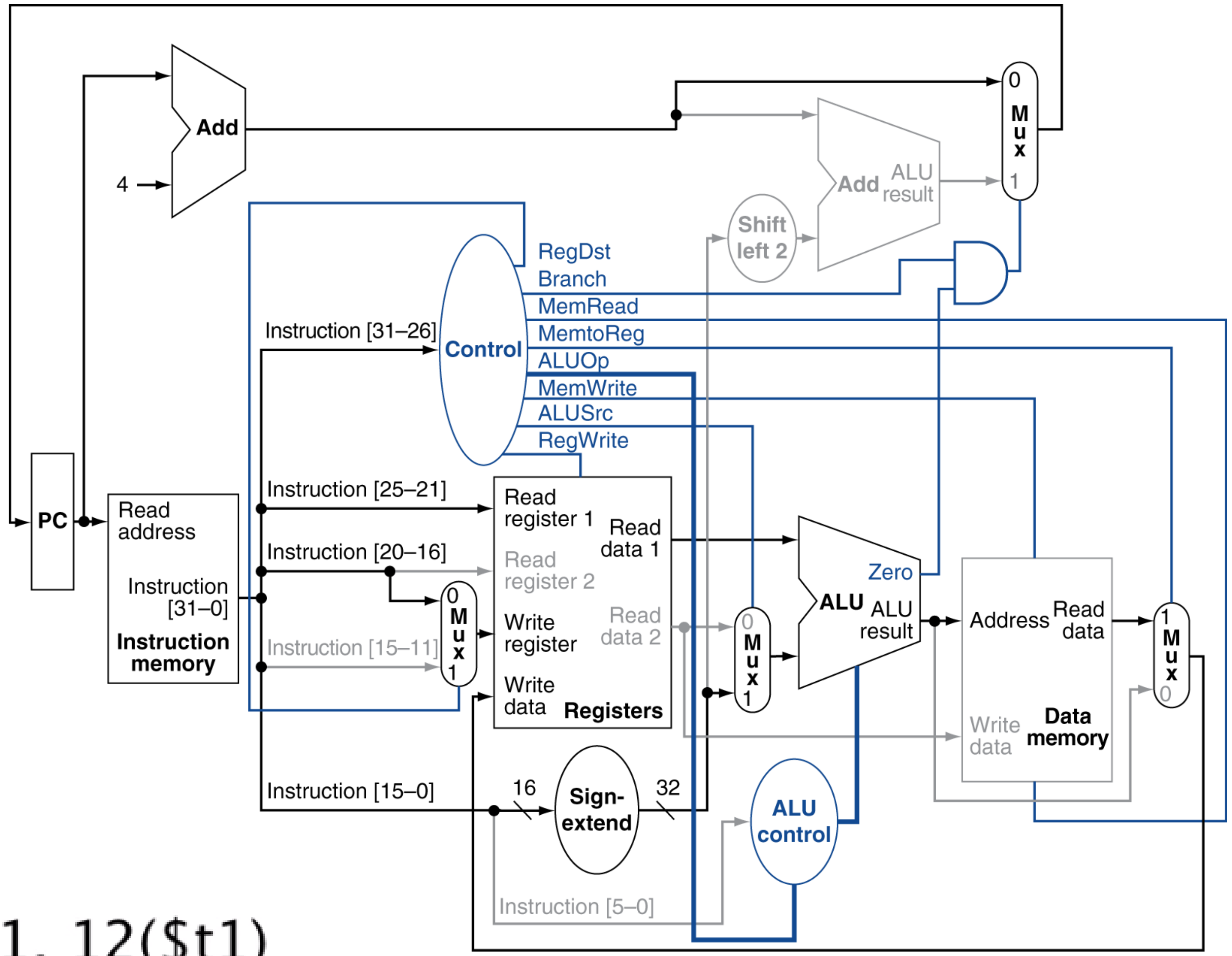
R-Type Instruction



rd rs rt

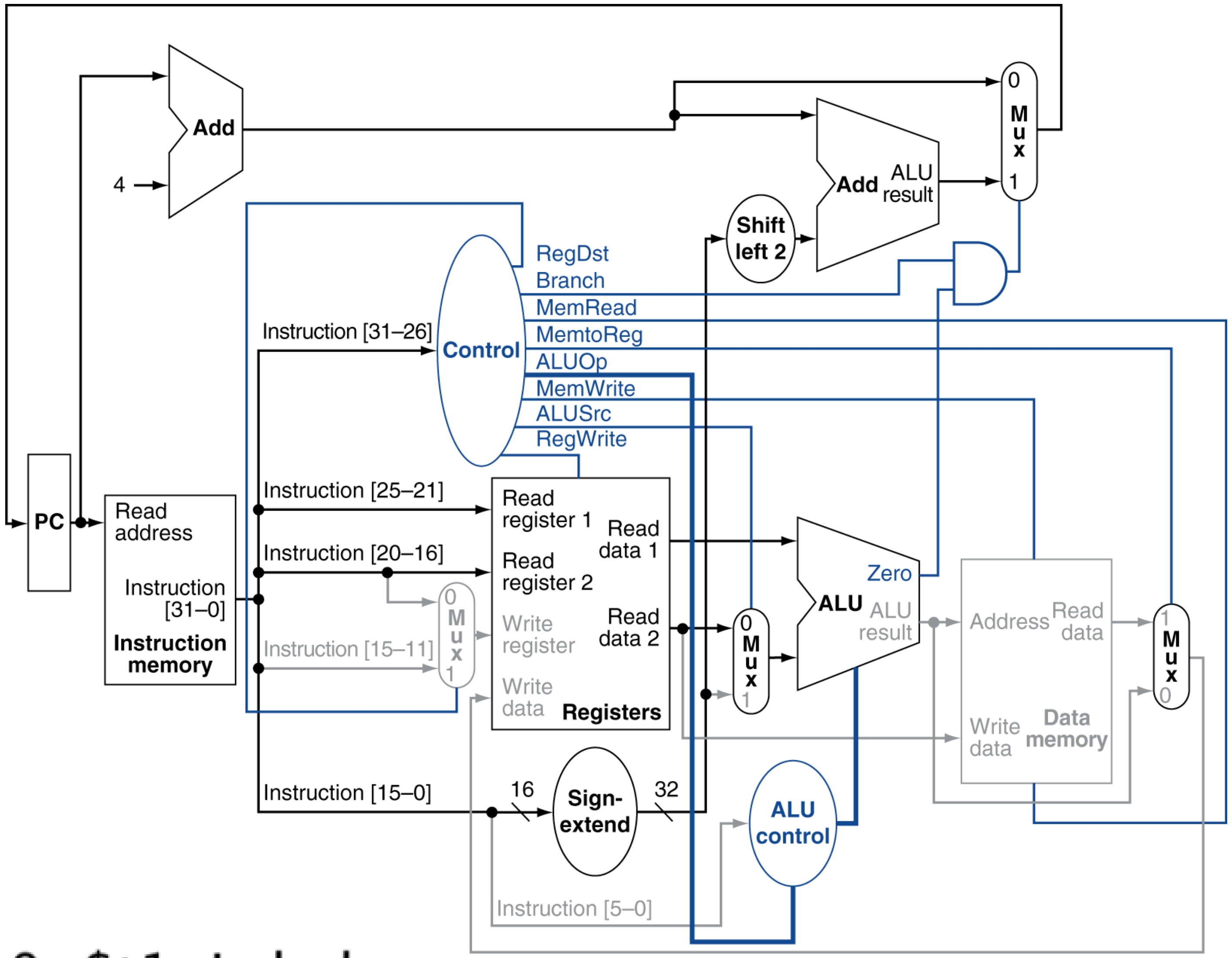
add $t1, $s1, $s2

Load Instruction



lw $s1, 12($t1)

Branch-on-Equal Instruction



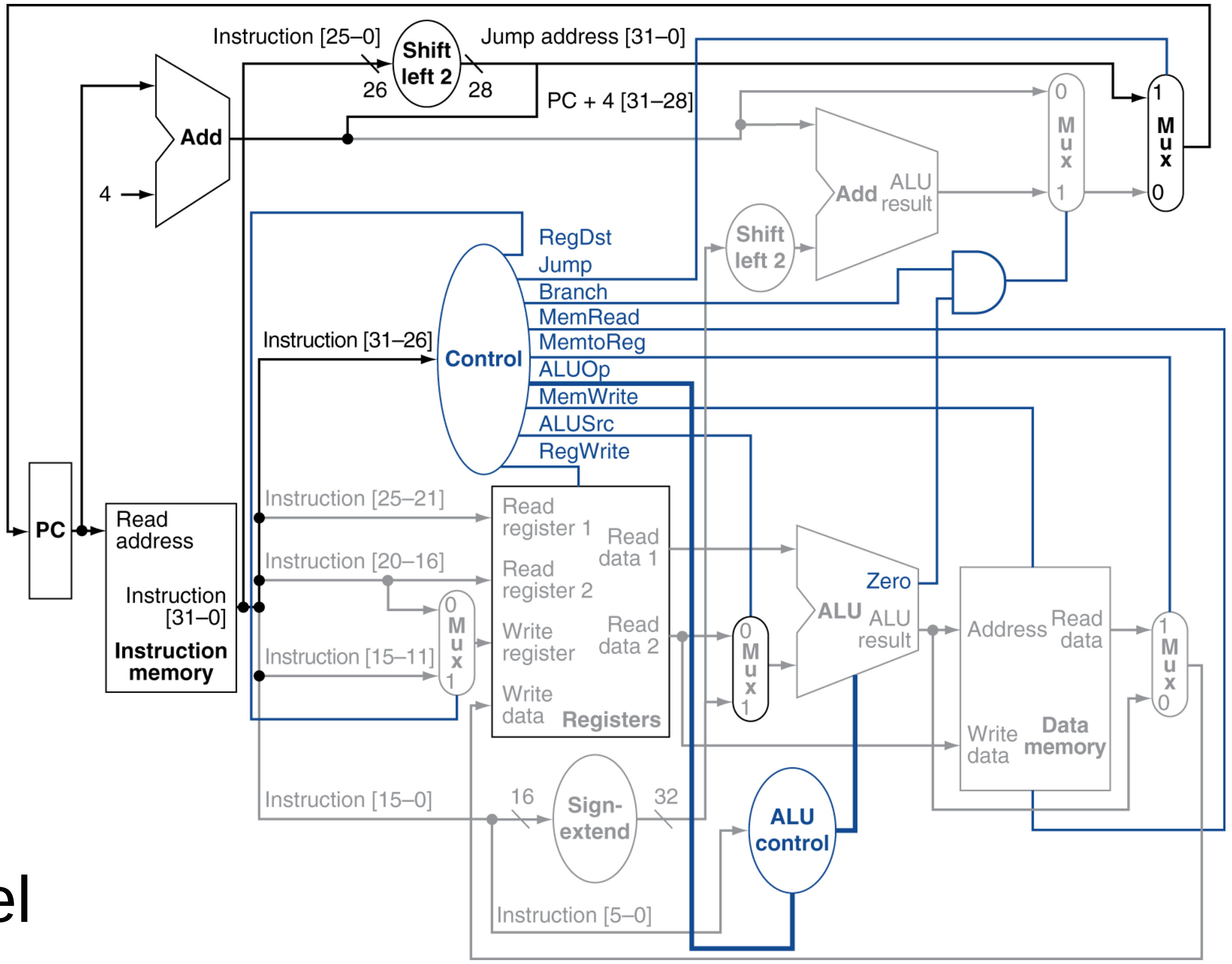
beq $t0, $t1, Label

Implementing Jumps

Jump	2	address
	31:26	25:0

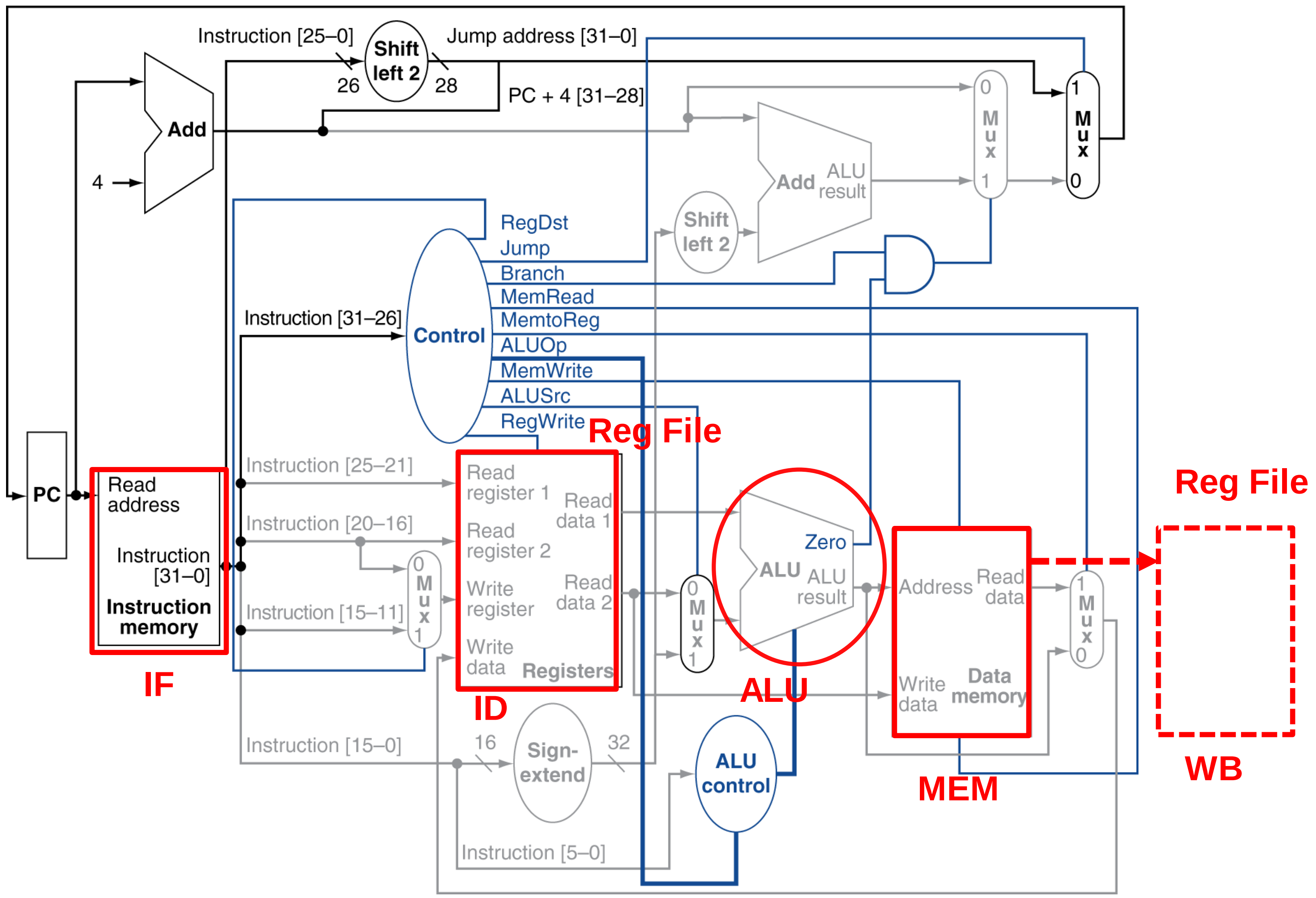
- Jump uses word address
- Update PC with concatenation of
  - Top 4 bits of old PC
  - 26-bit jump address
  - 00
- Need an extra control signal decoded from opcode

Datapath With Jumps Added



j Label

Datapath Overview



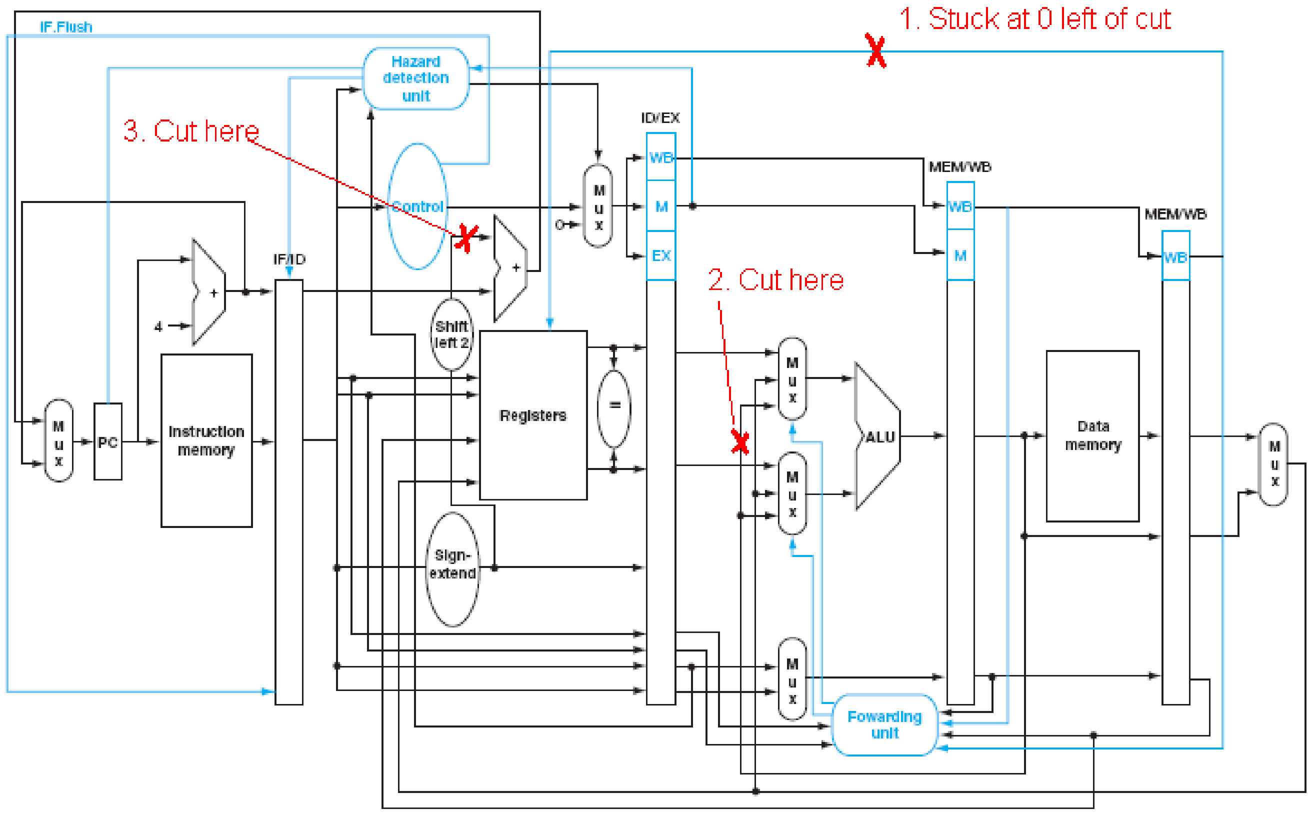
Performance Issues

- Longest delay determines clock period
  - Critical path: load instruction
  - Instruction memory $\rightarrow$ register file $\rightarrow$ ALU $\rightarrow$ data memory $\rightarrow$ register file
- Not feasible to vary period for different instructions
- Violates design principle
  - Making the common case fast
- We will improve performance by pipelining

Exersice

For the MIPS datapath shown below, several lines are marked with "X". For each one:

- Describe in words the negative consequence of cutting this line relative to the working, unmodified processor.
- Provide a snippet of code that will fail
- Provide a snippet of code that will still work



Solution

(1) Cannot write to register file. This means that R-type and any instruction with write back to register file will fail. An example of code snippet that would fail is:

```
add $s1, $s2, $s3
```

An example of a code snippet that will not fail is:

```
sw $s1, 0($s2)
```

(2) Forwarding of the first operand fails. An example of code snippet that would fail is:

```
add $s1, $t0, $t1
add $s1, $s1, $s1
```

An example of code snippet that will not fail is:

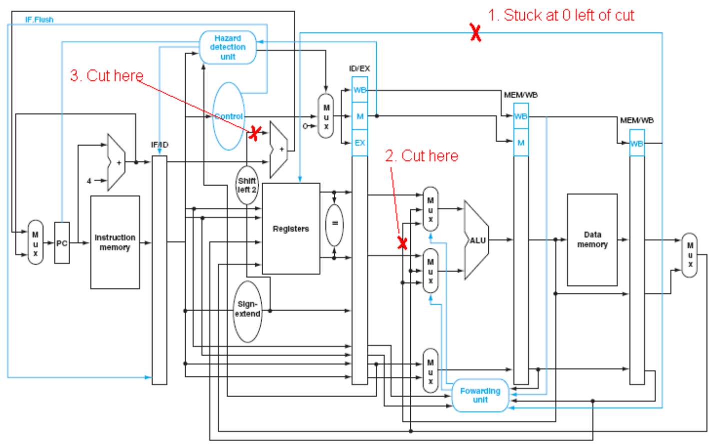
```
add $s1, $t0, $t1
add $s1, $t2, $s1 # Here the second operand is forwarded correctly
```

(3) Jumping to a branch target does not work. Example of code that fails:

```
addi $s1, $zero, 2
addi $s2, $zero, 2
beq $s1, $s2, exit
```

Code that will still work:

```
addi $s1, $zero, 10
addi $s2, $zero, 20
beq $s1, $s2, exit
```



1. Stuck at 0 left of cut
IF.Flush
Hazard detection unit
3. Cut here
Control
ID/EX
WB
M
EX
MEM/WB
WB
M
MEM/WB
WB
2. Cut here
IF/ID
Shift left 2
Registers
=
Instruction memory
PC
Mux
4
ALU
Data memory
Sign-extend
Fowarding unit
0