COMPUTER ORGANIZATION

CPU DESIGN

The Single-Cycle Implementation

PROF H ROUMANI
Dept. of Electrical Engineering and Computer Science, York University

1

CPU DESIGN

- The Datapath
- Single-Cycle Control
- Performance

Focus on the Subset:

`addi, add/sub/and/or/slt, lw/sw, beq, j`

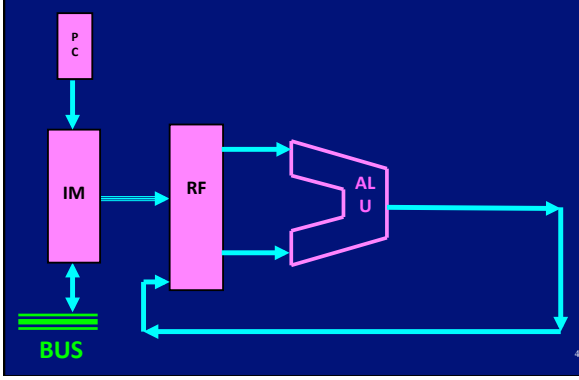
2

Building the

Datapath

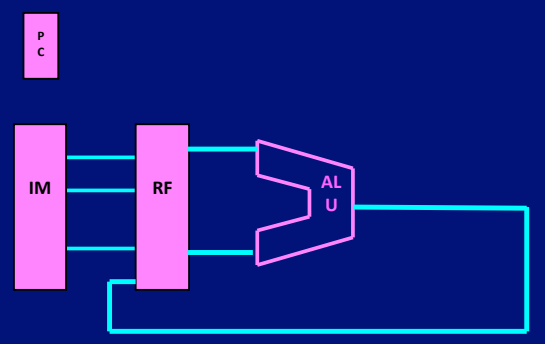
3

The Basic Datapath Components



4

The Basic Datapath [computational R-Type]



5

Recall the M/L Formats:

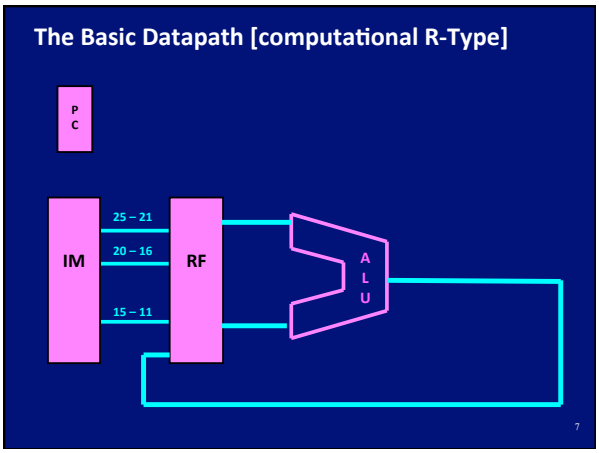
	6	5	5	5	5	6
R	opCode	rs	rt	rd	sa	funCode

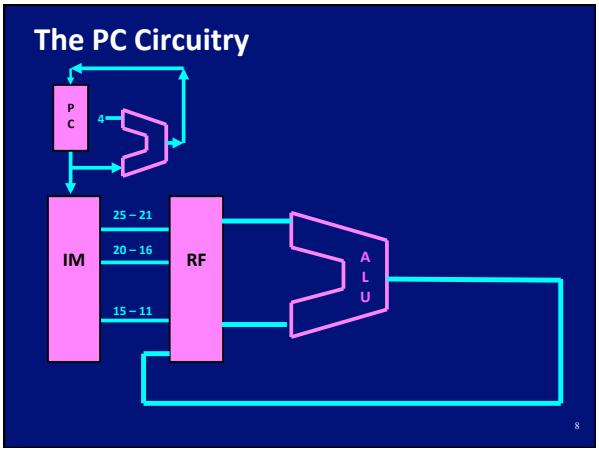
	6	5	5	16
I	opCode	rs	rt	immediate

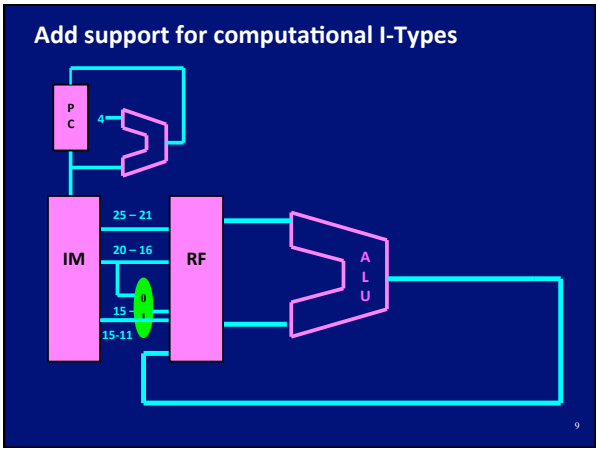
	6	26
J	opCode	immediate

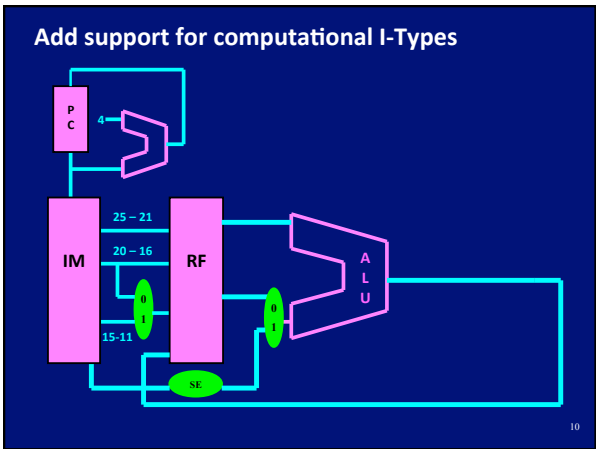
Register rs = source, rt = target, rd = destination.

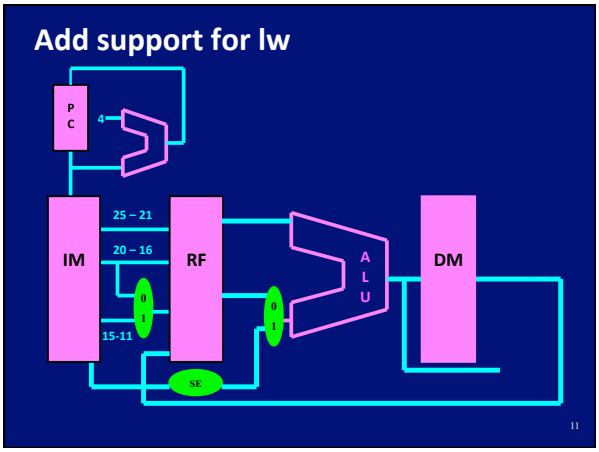
6

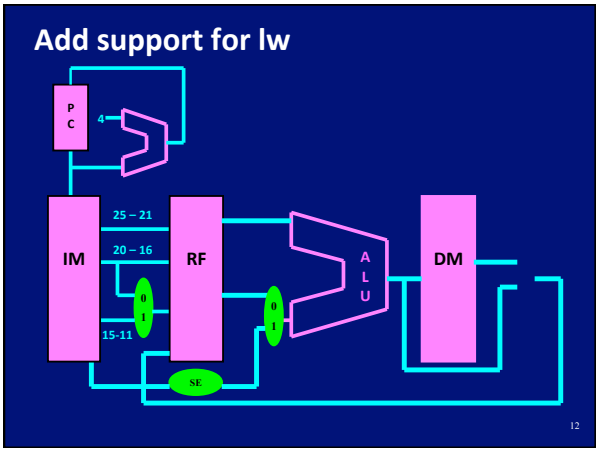

The Basic Datapath [computational R-Type]
P
C
IM
25 – 21
20 – 16
15 – 11
RF
A
L
U
7

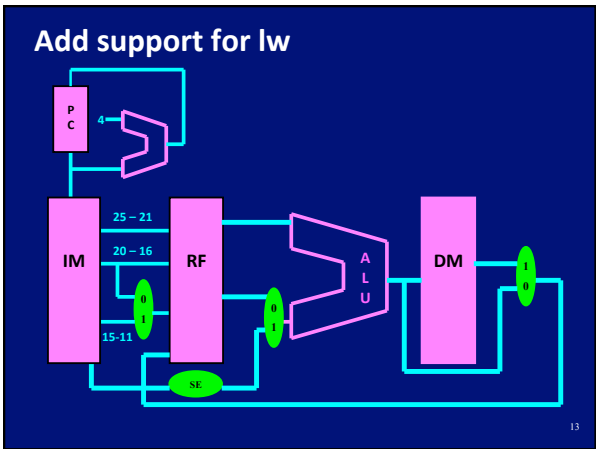

The PC Circuitry
P
C
4
IM
25 – 21
20 – 16
15 – 11
RF
A
L
U
8

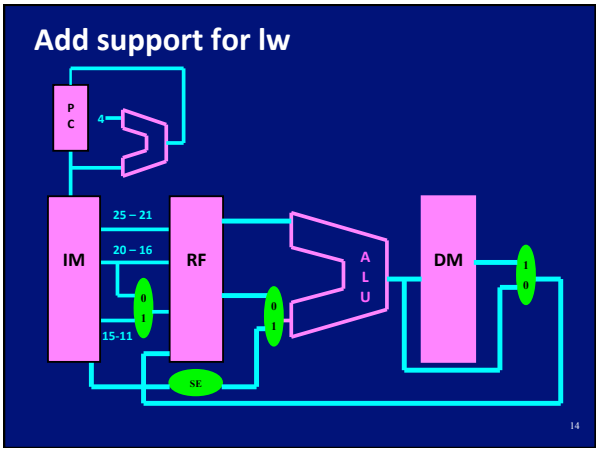

Add support for computational I-Types
P
C
4
IM
25 – 21
20 – 16
15 –
15-11
0
RF
A
L
U
9

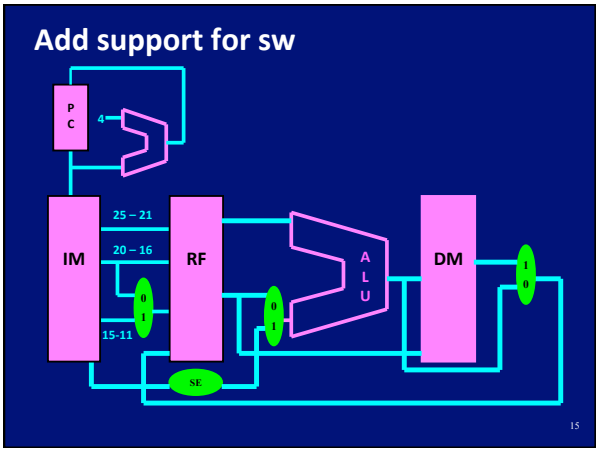

Add support for computational I-Types
P
C
4
IM
25 – 21
20 – 16
15-11
RF
A
L
U
SE
10

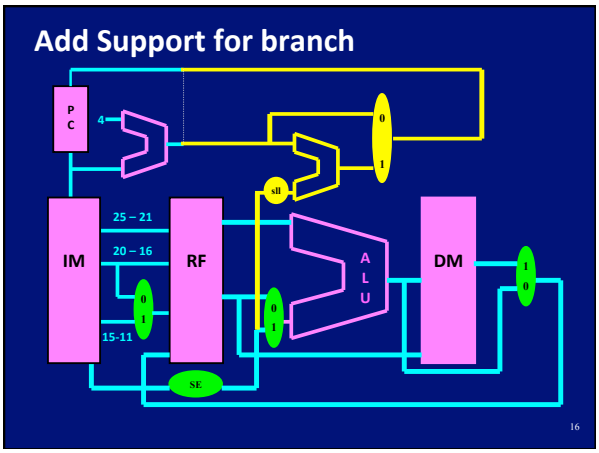

Add support for lw
P
C
4
IM
25 – 21
20 – 16
15-11
RF
A
L
U
DM
SE
11

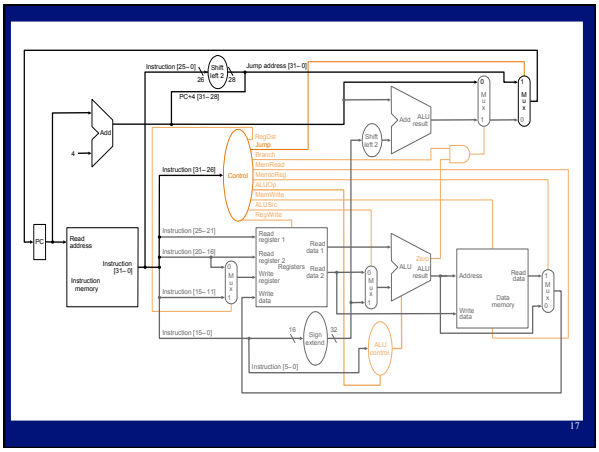

Add support for lw
P
C
4
IM
25 – 21
20 – 16
15-11
RF
A
L
U
DM
SE
12


Add support for lw
P
C
4
IM
25 – 21
20 – 16
15-11
RF
A
L
U
DM
SE
13


Add support for lw
P
C
4
IM
25 – 21
20 – 16
15-11
RF
A
L
U
DM
SE
14

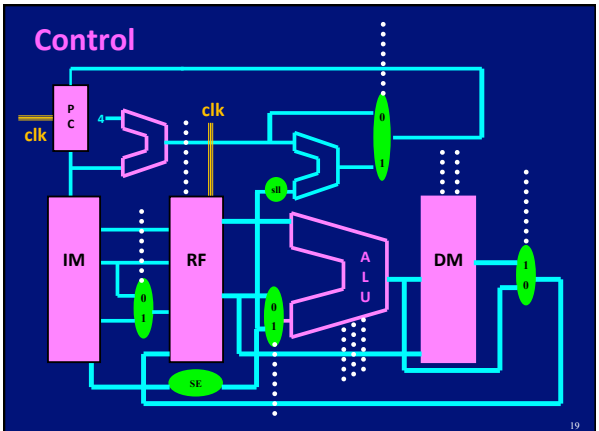

Add support for sw
P
C
4
IM
25 – 21
20 – 16
15-11
RF
A
L
U
DM
SE
15


Add Support for branch
P C
4
0
1
sll
IM
25 – 21
20 – 16
15-11
RF
A L U
DM
SE
16


Add
4
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–0]
Shift left 2
Jump address [31–0]
PC+4 [31–28]
Instruction [31–26]
Control
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
16
Sign extend
32
Shift left 2
Add
ALU result
ALU
Zero
ALU control
Address
Read data
Data memory
Write data
M u x
17


Building the
Control
18

Control



19

Exercise

```
add $t0, $s0, $a0
```

SIGNAL	VALUE
ALUSrc	0
MemToReg	0
RegDst	1
RegWrite	1
MemRead	0
MemWrite	0
Branch	0
Jump	0
Operation (3-bit)	*addition*

20

Exercise

```
sw $t0, 500($s0)
```

SIGNAL	VALUE
ALUSrc	1
MemToReg	d/c
RegDst	d/c
RegWrite	0
MemRead	0
MemWrite	1
Branch	0
Jump	0
Operation (3-bit)	

21

Exercise

```
beq $t0, $s0, 401
```

SIGNAL	VALUE
ALUSrc	0
MemToReg	d/c
RegDst	d/c
RegWrite	0
MemRead	0
MemWrite	0
Branch	1
Jump	0
Operation (3-bit)	

22

Generating the Control Signals

All signals depend on the instruction, i.e. on a total of 12 bits ➔ **complex**.

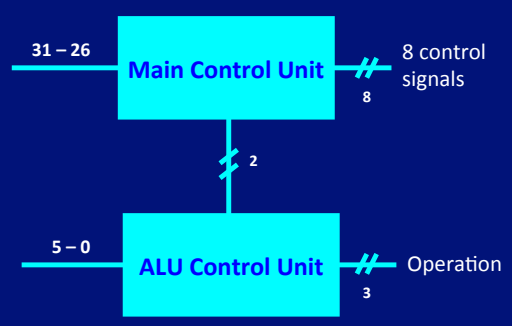
Note that non-ALU signals depend only on the 6-bit op_code ➔ **simpler**.

Hence, **split** the control into a main control unit that sees only the opcode, and an **auxiliary** one that sees the funtion code.

The two communicate via a new signal, ALUop

23

Splitting the Control



24

The Operation Signal

A 3-bit signal through which the auxiliary control unit tells the ALU to:

```
000 = and
001 = or
010 = add
110 = sub
111 = slt
```

25

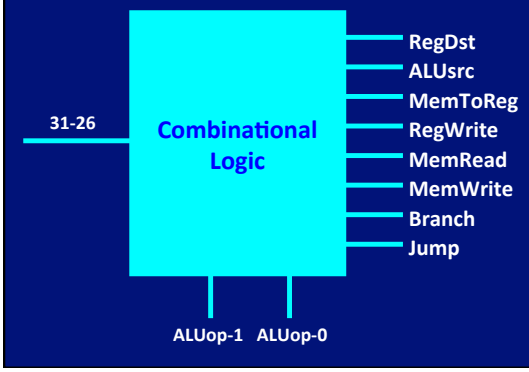
The ALUop Signal

A 2-bit signal through which the main control unit tells the auxiliary to:

- 00 = add (no matter what the fun_code is)
- 01 = subtract (no matter what the fun_code is)
- 10 = R-Type (follow the fun_code

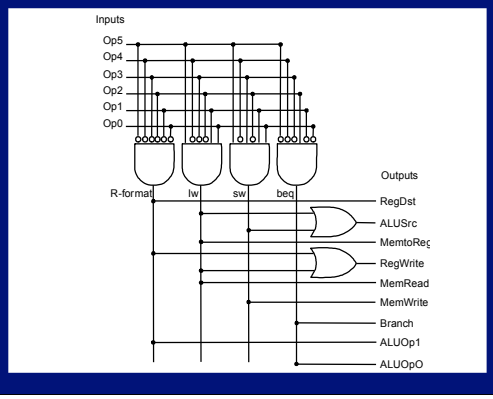
26

The Main Control Unit



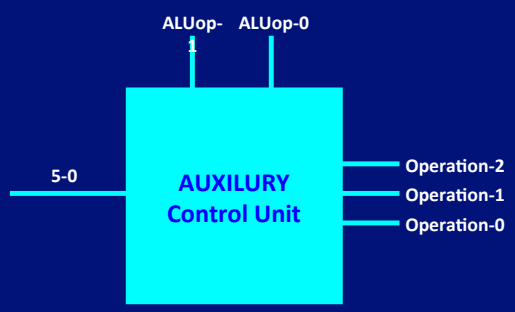
27

The Main Control Unit



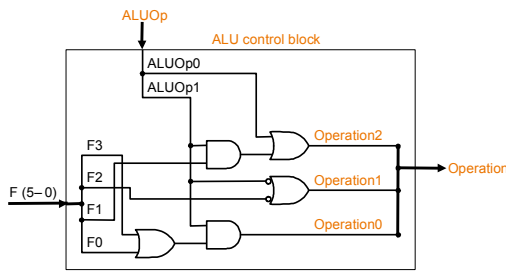
28

ALU Control



29

Aux Controller Implementation



30

The Single-Cycle

Performance

31

Component Delays
RF=50, ALU=100, and MEM (both IM and DM)=200 ps.

Compute CPU Time to execute various instructions
j, beq, add, sw, lw

Compute Max GHz for the CPU Clock
Answer: 1.66 GHz

Critique of S/Cycle
+very simple
-caters to the slowest
-h/w redundancy

32