INTRO. TO COMP. ENG. CHAPTER XII-1 SINGLE CYCLE DPU **•CHAPTER XII** 

# **CHAPTER XII**

# SINGLE CYCLE DATAPATH UNIT

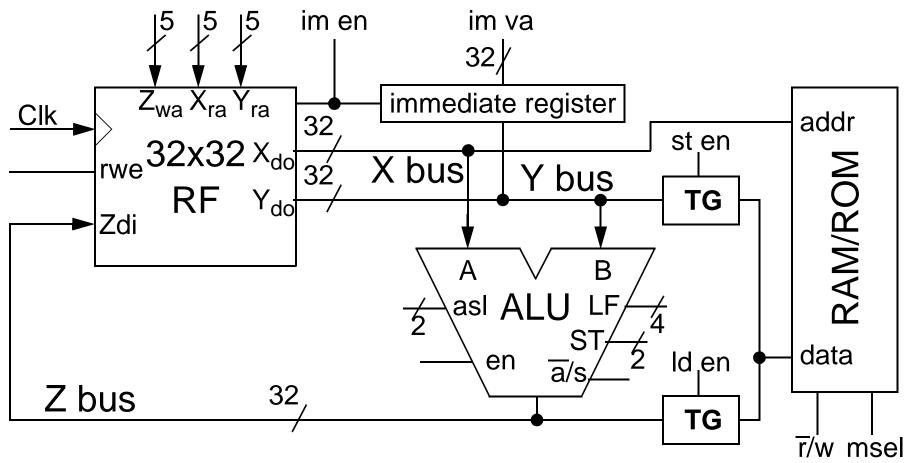
READ SINGLE CYCLE DATAPATH FREE-DOC ON COURSE WEBPAGE

## INTRO. TO COMP. ENG. CHAPTER XII-14 SINGLE CYCLE DPU

# SINGLE CYCLE DPU

#### **INCLUDING MEMORY**

- •SINGLE CYCLE DPU
  - -ARITHMETIC LOGIC UNIT
    - -SINGLE CYCLE DPU W/ALU
  - -IMMEDIATE REGISTER
- 32x32 bits is not sufficient memory for most computers.
- We can include external memory (SRAM, DRAM, etc.) as follows.



# INTRO. TO COMP. ENG. CHAPTER XI-6

**DATAPATH ELEMENTS** 

## REGISTER FILES

32-BIT WORD, 32 REGISTERS

•REGISTER FILES
-REGISTER LAYOUT
-WRITE DECODER

-READ DECODER

• For the upcoming datapath designs in the next chapter, we want to have a 32x32 register file with one write input and two read outputs.

X<sub>ra</sub> - X read address

Y<sub>ra</sub> - Y read address

Z<sub>wa</sub> - Z write address

X<sub>do</sub> - X data out

Y<sub>do</sub> - Y data out

Z<sub>di</sub> - Z data in

rwe - register write enable

Clk  $Z_{wa}$   $Z_{wa}$   $Z_{wa}$   $Z_{wa}$   $Z_{wa}$   $Z_{wa}$   $Z_{va}$   $Z_{va}$ 

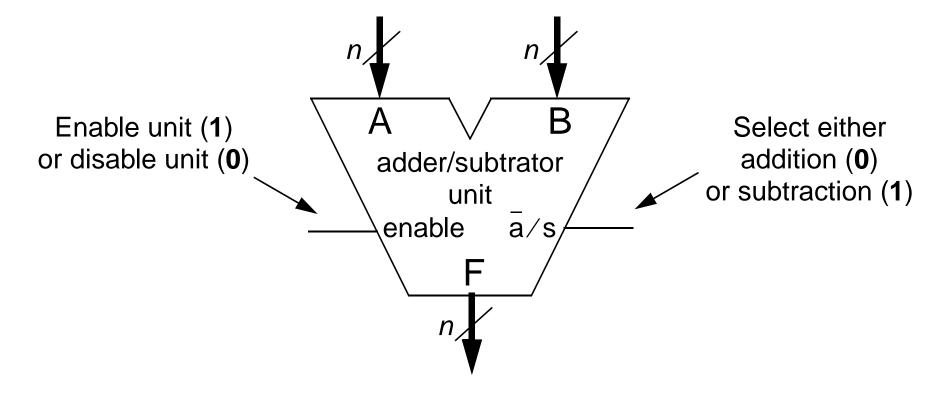
Note: Two data outputs implemented with two read address decoders.

## INTRO. TO COMP. ENG. CHAPTER XI-7 DATAPATH ELEMENTS

# ADDER/SUBTRACTOR

GENERAL UNIT DIAGRAM

- •REGISTER FILES
  - -WRITE DECODER
  - -READ DECODER
  - -32X32 REGISTER FILE
- An n-bit adder/subtractor unit is often illustrated as follows.



This unit would have n full-adders internally.

#### INTRO. TO COMP. ENG. CHAPTER XI-9 DATAPATH ELEMENTS

# **LOGICAL UNIT**

#### INTRODUCTION

•REGISTER FILES
•ADDER/SUBTRACTOR
-GENERAL UNIT DIAGRAM
-OTHER UNIT SIGNALS

- A useful unit would be one that can take two n-bit inputs and perform some logical operation between each of the bits to get an n-bit output.
  - For example, given the 8-bit values **0001 1110** and **1001 1000**, we might want to find the **bit-wise logical OR**.

bit-wise 0001 1110 1001 1000 1001 1110

Or similarly, the bit-wise logical AND of the two 8-bit values.

bit-wise 0001 1110 1001 1000 0001 1000

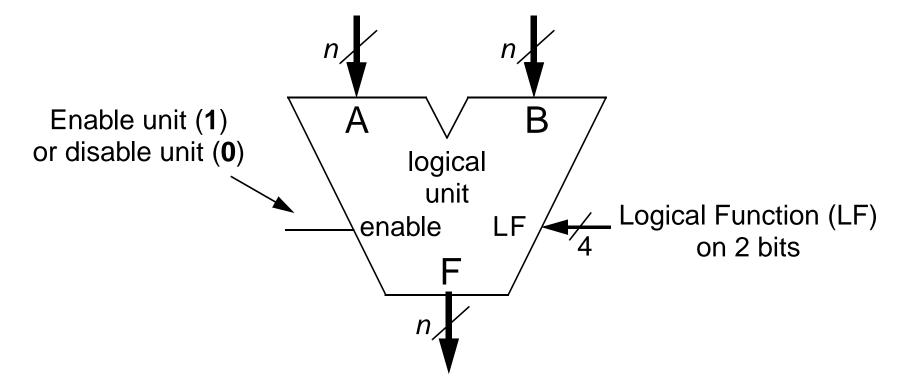
These types of operations are often used for masking and setting bits.

## INTRO. TO COMP. ENG. CHAPTER XI-10 DATAPATH ELEMENTS

# **LOGICAL UNIT**

#### GENERAL UNIT DIAGRAM

- •REGISTER FILES
  •ADDER/SUBTRACTOR
  •LOGICAL UNIT
  -INTRODUCTION
- Below is a general unit diagram for an n-bit logical unit.



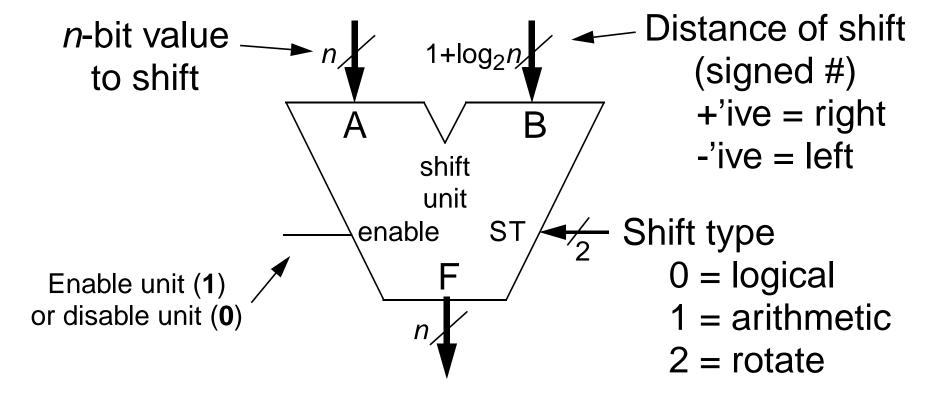
 Logical operations, such as AND/OR/NOT/NAND/NOR/etc., are done for each bit of A and B to form F.

#### INTRO. TO COMP. ENG. CHAPTER XI-15 DATAPATH ELEMENTS

# **SHIFT UNIT**

#### GENERAL UNIT DIAGRAM

- •LOGICAL UNIT
  •SHIFT UNIT
  -INTRODUCTION
- Below is a general unit diagram for an n-bit shift unit.



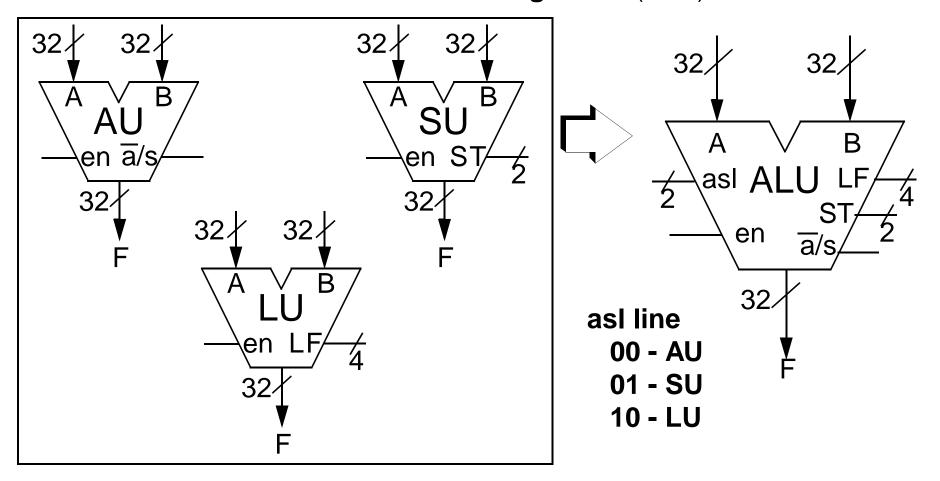
 Notice that the n-bit value A will be shifted according to the distance indicated with signed number B.

#### INTRO. TO COMP. ENG. CHAPTER XII-10 SINGLE CYCLE DPU

## SINGLE CYCLE DPU

#### ARITHMETIC LOGIC UNIT

- **•SINGLE CYCLE DPU** 
  - -ADD/SUBTRACT MACHINE
  - -BASIC SINGLE CYCLE DPU
  - -COMPUTATION EXAMPLES
- Since only one of **AU**, **SU**, or **LU** will be active at a time in this architecture, we will combine to form an **arithmetic logic unit** (ALU).

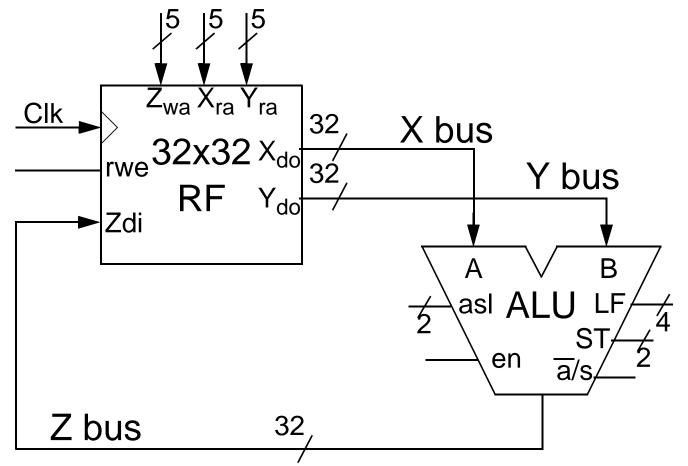


#### INTRO. TO COMP. ENG. CHAPTER XII-11 SINGLE CYCLE DPU

# SINGLE CYCLE DPU

SINGLE CYCLE DPU W/ALU

- **•SINGLE CYCLE DPU** 
  - -BASIC SINGLE CYCLE DPU
  - -COMPUTATION EXAMPLES
  - -ARITHMETIC LOGIC UNIT
- Using our ALU, the DPU can be redrawn as follows.



• This structure is still a **triple bus internal DPU architecture**.

#### INTRO. TO COMP. ENG. CHAPTER XII-13 SINGLE CYCLE DPU

# SINGLE CYCLE DPU

#### IMMEDIATE REGISTER

- •SINGLE CYCLE DPU
  - -ARITHMETIC LOGIC UNIT
  - -SINGLE CYCLE DPU W/ALU
  - -IMMEDIATE REGISTER

- The im\_en line does two things:
  - When **0**, **im\_en** controls
    - immediate register outputs to go to high impedence so as NOT to affect Y bus.
    - register file Y data out to output corresponding register value.
  - When 1, im\_en controls
    - **immediate register** to output register value to **Y bus**.
    - register file Y data out to go to high impedence so as NOT to affect Y bus.
- The im\_va lines pass a value to the immediate register.

## INTRO. TO COMP. ENG. CHAPTER XII-18 SINGLE CYCLE DPU

# SINGLE CYCLE DPU

#### **MICROCODE**

•SINGLE CYCLE DPU

- -INCLUDING MEMORY
- -READING FROM MEMORY
- -WRITING TO MEMORY
- Microcode in a processor are all of the control signals required to execute an operation for a clock cycle.
- We have actually looked at examples of a microcode operation when we considered various operations such as

$$R3 = R1 + R2$$

or

$$M[R5] = R9$$

 Later we will talk about macrocode which are longer operations consisting of many microcode operation over a number of clock cycles.

#### INTRO. TO COMP. ENG. CHAPTER XII-16 SINGLE CYCLE DPU

# SINGLE CYCLE DPU

#### READING FROM MEMORY

- SINGLE CYCLE DPU
  - -SINGLE CYCLE DPU W/ALU
  - -IMMEDIATE REGISTER
  - -INCLUDING MEMORY
- We wish to be able to read and write from our memory.
- A sample read/load operation can be expressed as follows

$$R4 = M[R7]$$

- This operation uses the value in R7 as the address to the memory and reads the value at that address in the memory to R4.
- What control signals are required?
  - en = 0 for ALU.
  - $X_{ra} = 00111$ ,  $Y_{ra} = XXXXX$ ,  $Z_{wa} = 00100$ , and rwe = 1 for RF.
  - st\_en = 0 and Id\_en = 1
  - ~r/w = r and msel = 1

#### INTRO. TO COMP. ENG. CHAPTER XII-17 SINGLE CYCLE DPU

# SINGLE CYCLE DPU

#### WRITING TO MEMORY

- •SINGLE CYCLE DPU
  - -IMMEDIATE REGISTER
  - -INCLUDING MEMORY
  - -READING FROM MEMORY
- A sample write/store operation can be expressed as follows

$$M[R5] = R9$$

- This operation uses the value in R5 as the address to the memory and write the value in R9 to that address in the memory.
- What control signals are required?
  - en = 0 for ALU.
  - $X_{ra} = 00101$ ,  $Y_{ra} = 01001$ ,  $Z_{wa} = XXXXX$ , and rwe = 0 for RF.
  - st\_en = 1 and Id\_en = 0
  - ~r/w = w and msel = 1

INTRO. TO COMP. ENG. CHAPTER XIII-3 ISA

## **PROGRAM PATH**

TRANSLATING CODE

•ISA
-INTRODUCTION

Below is the process for translating a program to machine opcodes.

**Compiler translates** program add \$10, \$8, \$9 xor \$13, \$11, \$12 Iw \$15,0(\$16) Assembler converts to machine code 010110001010111101001001 10010101001101011101101 00101110100101010111011

High level program e.g. C, C++,
Pascal, Java

Assembly language program

**Machine instructions** 

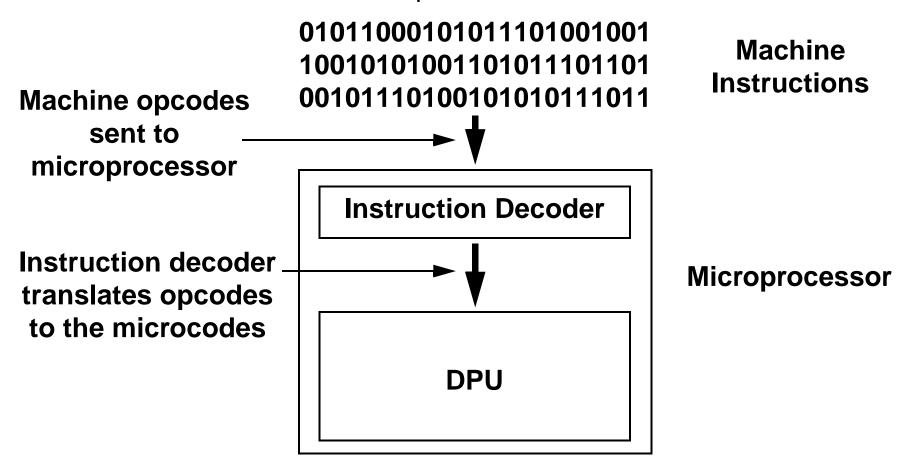
INTRO. TO COMP. ENG. CHAPTER XIII-4 ISA

## **PROGRAM PATH**

**EXECUTING CODE** 

•ISA
•PROGRAM PATH
-TRANSLATING CODE

 Once the opcodes are given to the microprocessor, it translates the opcode instructions to the microcodes operations we discussed.



## INTRO. TO COMP. ENG. CHAPTER XIII-18 ISA

# **INSTRUCTIONS**

#### **INSTRUCTION FORMATS**

- •MIPS ASSEMBLY
  - -REGISTER INSTRUCTIONS
  - -IMMEDIATE INST. FORMAT
  - -IMMEDIATE INSTRUCTIONS
- While instructions can come in many different shapes and forms, we will consider the following 32-bit instruction formats to loosely follow the MIPS R3000/4000 format.

R-format	31 2 opcode	25 2 Z	<b>O</b> X	15 1 Y	other potential bits
31 25 20 15 0					
<b>I-format</b>	opcode	Z	X		immediate value

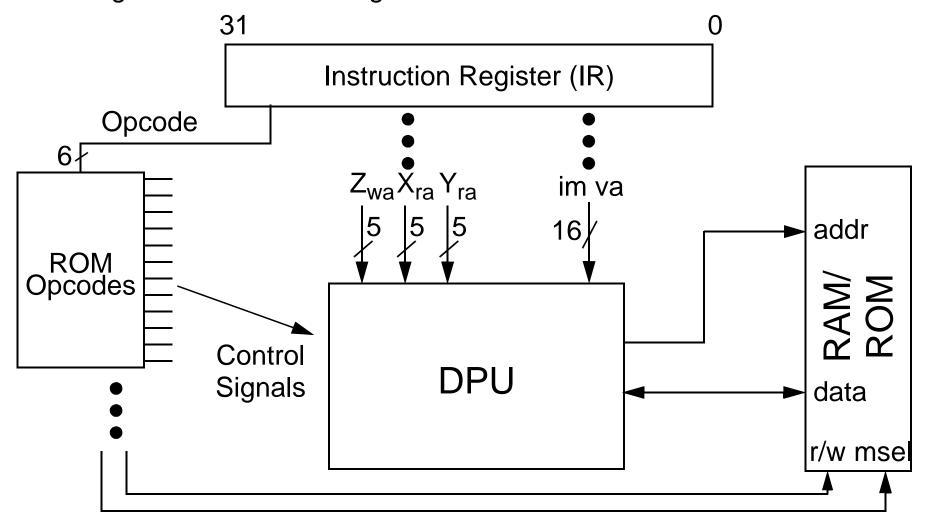
INTRO. TO COMP. ENG. CHAPTER XIII-21 ISA

## **INSTRUCTIONS**

#### INSTRUCTION REGISTER

- •MIPS ASSEMBLY
  - -INSTRUCTION FORMATS
  - -R-FORMAT W/ DPU
  - -I-FORMAT W/ DPU

• Use a general instruction register that can act as R- or I-Format.



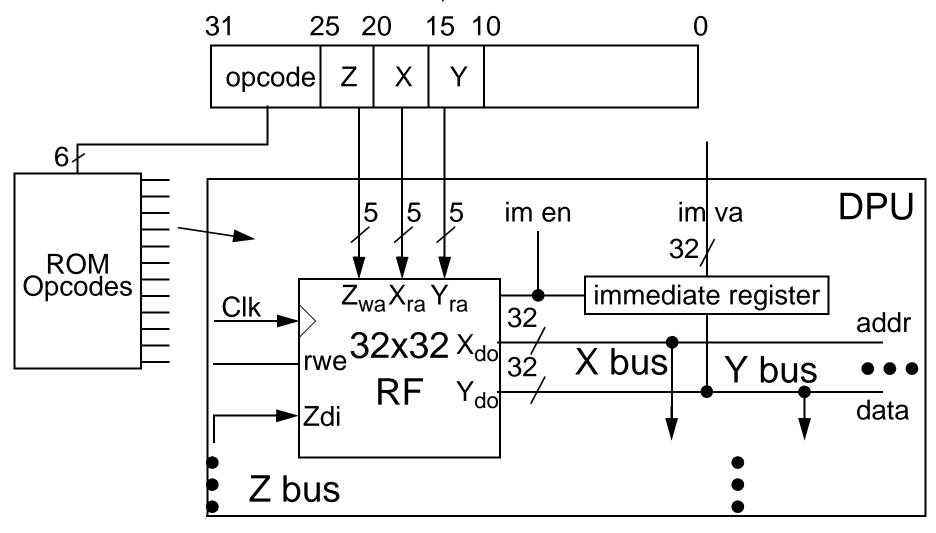
INTRO. TO COMP. ENG. CHAPTER XIII-19 ISA

# **INSTRUCTIONS**

R-FORMAT W/ DPU

- •MIPS ASSEMBLY
  - -IMMEDIATE INST. FORMAT
  - -IMMEDIATE INSTRUCTIONS
  - -INSTRUCTION FORMATS

If we have an R-format instruction, we link the bits as follows.



INTRO. TO COMP. ENG. CHAPTER XIII-20 ISA

# **INSTRUCTIONS**

I-FORMAT W/ DPU

- •MIPS ASSEMBLY
  - -IMMEDIATE INSTRUCTIONS
  - -INSTRUCTION FORMATS
  - -R-FORMAT W/ DPU

If we have an I-format instruction, we link the bits as follows.

