CHAPTER XII

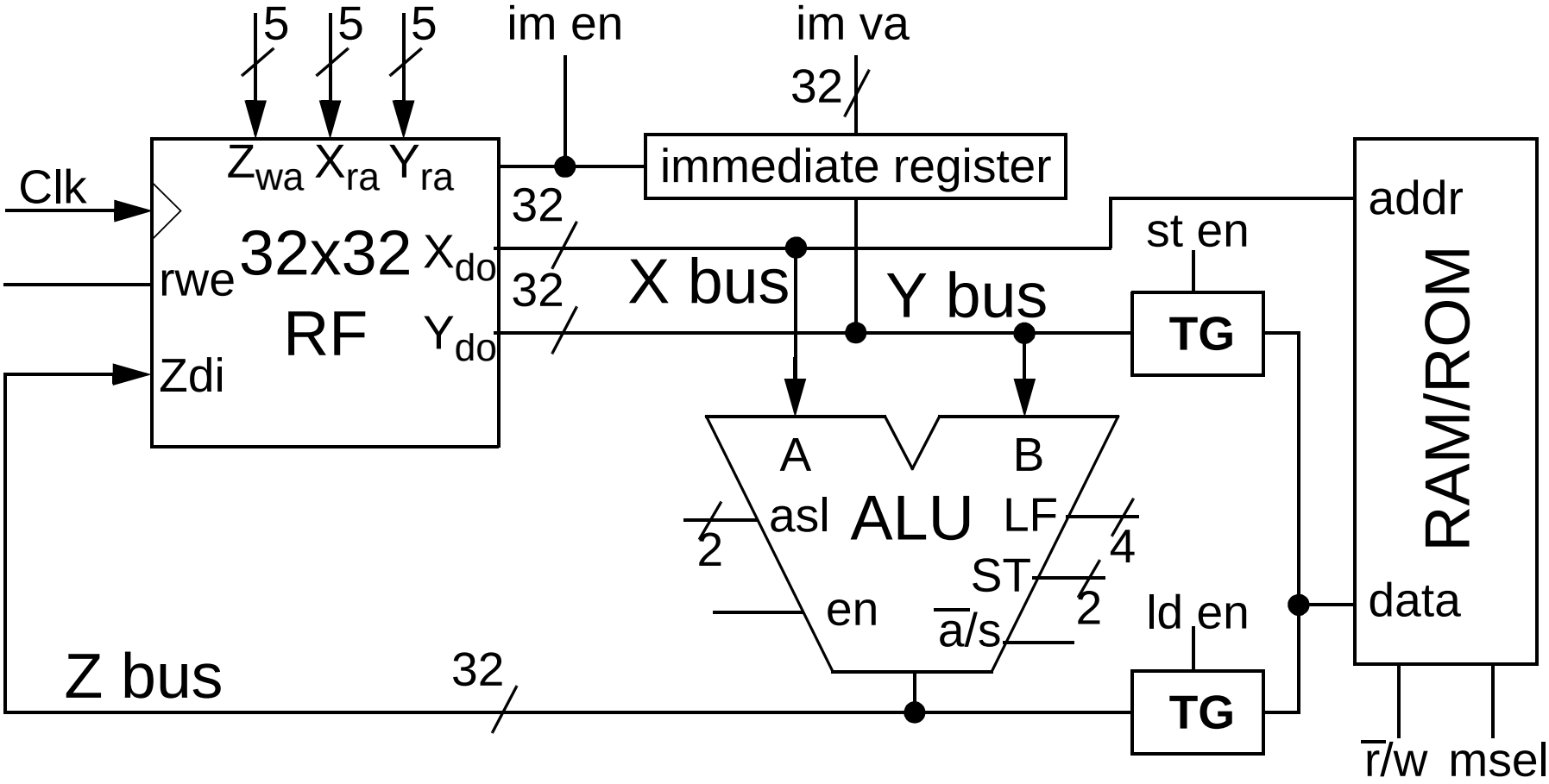
SINGLE CYCLE DATAPATH UNIT

READ SINGLE CYCLE DATAPATH FREE-DOC ON COURSE WEBPAGE

SINGLE CYCLE DPU

INCLUDING MEMORY

- 32x32 bits is not sufficient memory for most computers.
- We can include external memory (SRAM, DRAM, etc.) as follows.



REGISTER FILES

32-BIT WORD, 32 REGISTERS

- **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_{ra} - X read address

Y_{ra} - Y read address

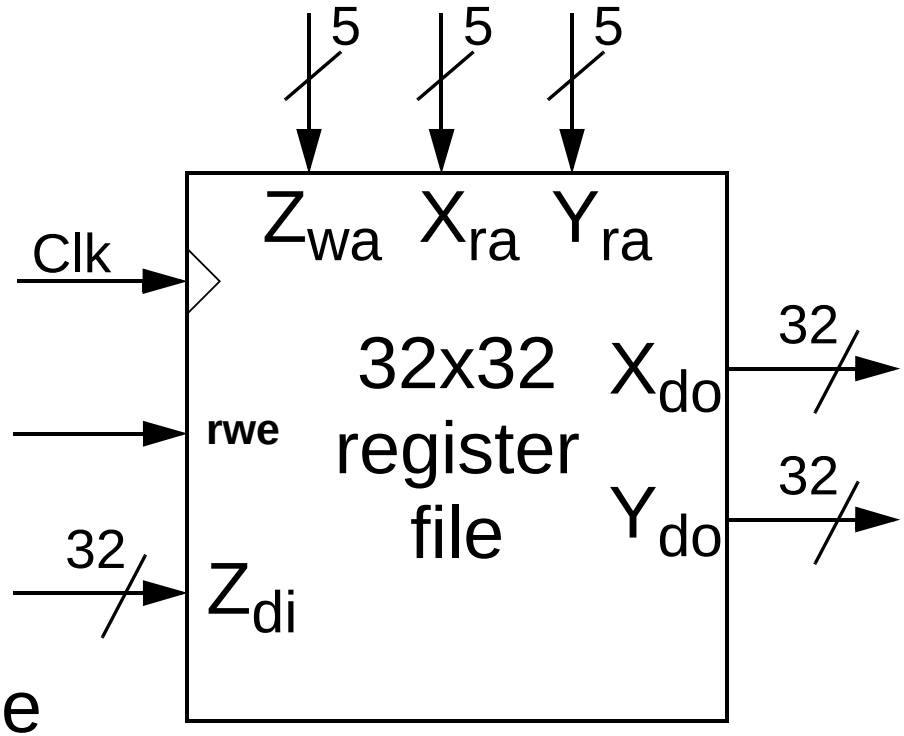
Z_{wa} - Z write address

X_{do} - X data out

Y_{do} - Y data out

Z_{di} - Z data in

rwe - register write enable

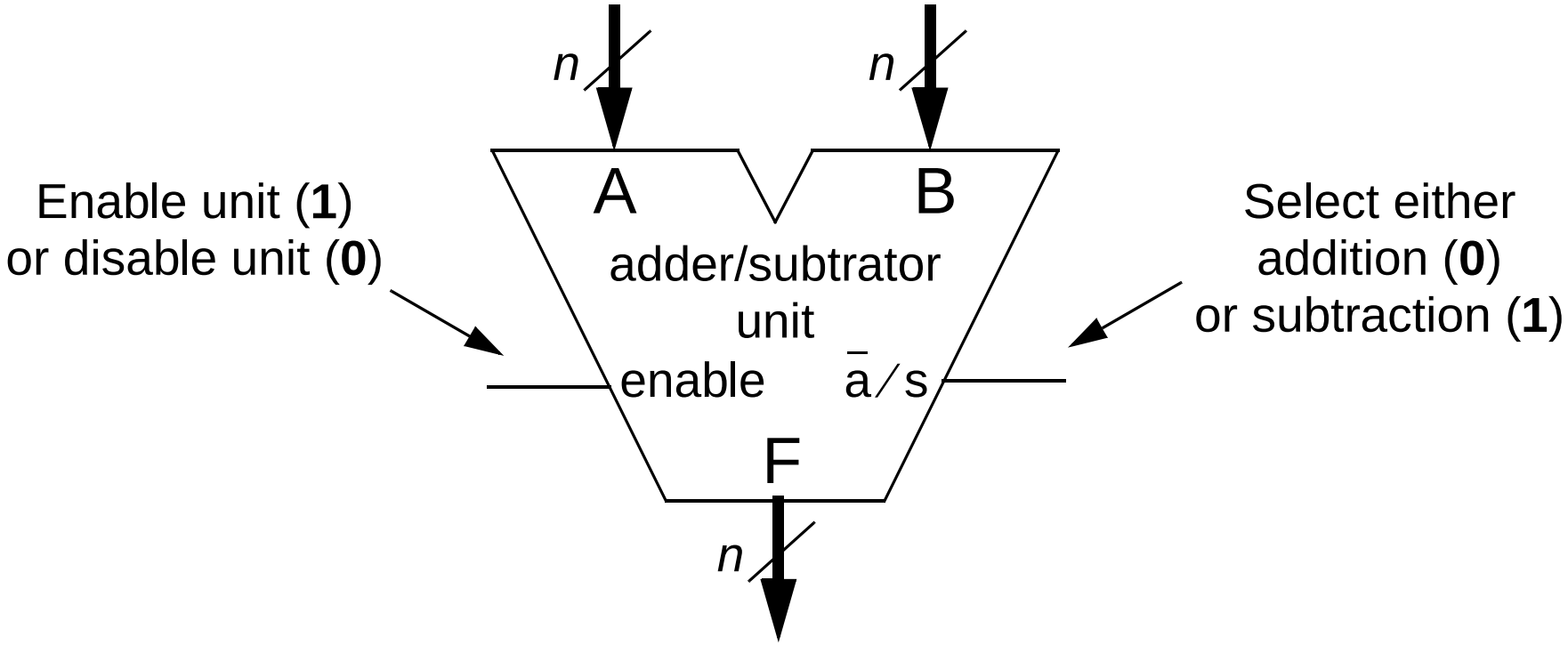


- Note: Two data outputs implemented with two read address decoders.

ADDER/SUBTRACTOR

GENERAL UNIT DIAGRAM

- **An *n*-bit adder/subtractor unit is often illustrated as follows.**



- This unit would have *n* full-adders internally.

LOGICAL UNIT

INTRODUCTION

- **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**.

	0001 1110
bit-wise logical OR	**1001 1000**
	1001 1110

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

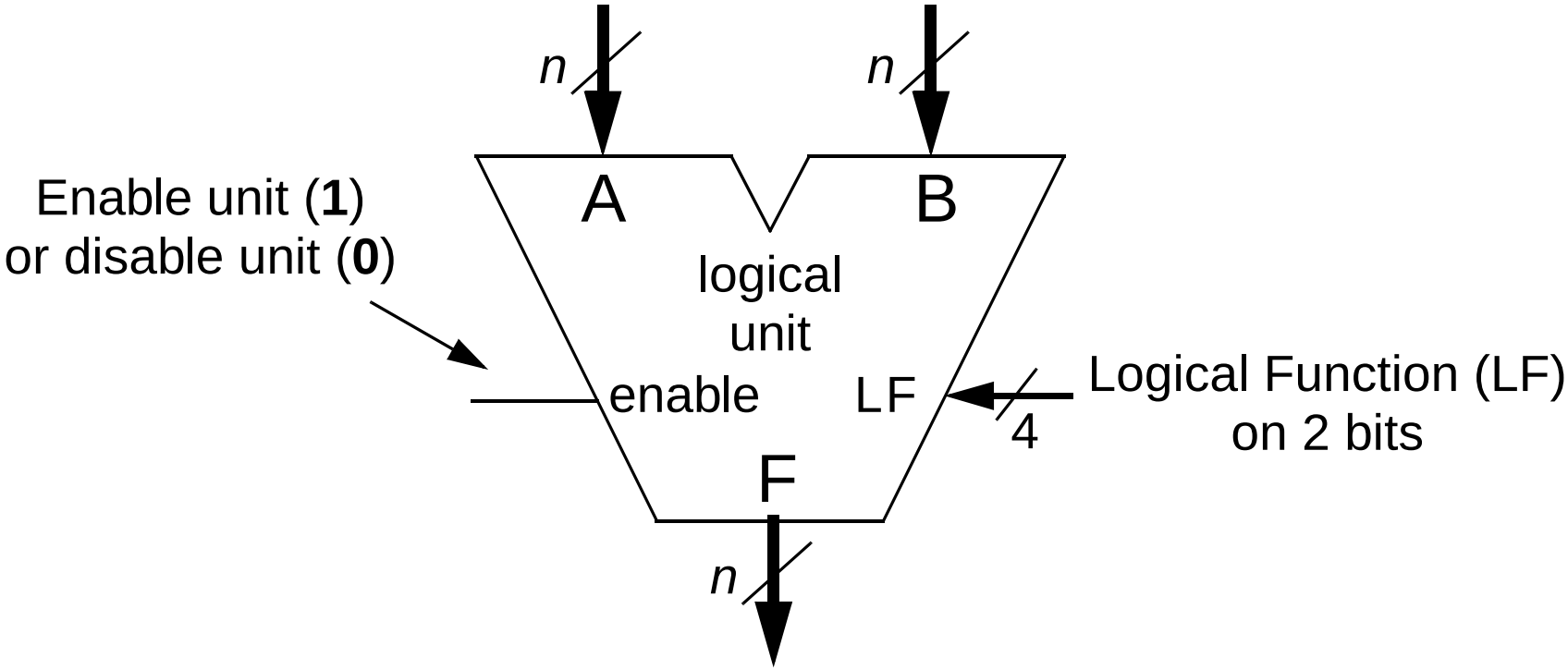
	0001 1110
bit-wise logical AND	**1001 1000**
	0001 1000

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

LOGICAL UNIT

GENERAL UNIT DIAGRAM

- **Below is a general unit diagram for an *n*-bit logical unit.**

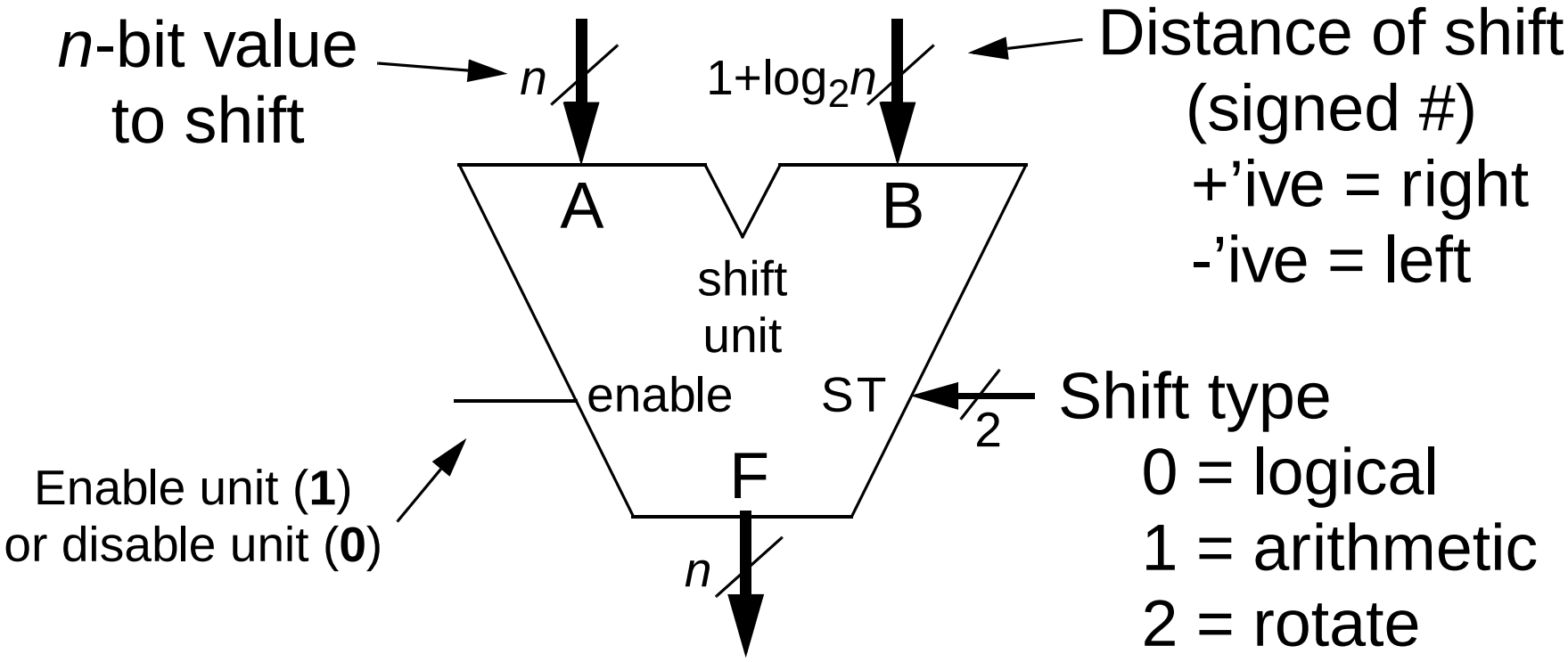


- Logical operations, such as **AND/OR/NOT/NAND/NOR**/etc., are done for each bit of **A** and **B** to form **F**.

SHIFT UNIT

GENERAL UNIT DIAGRAM

- **Below is a general unit diagram for an *n*-bit shift unit.**

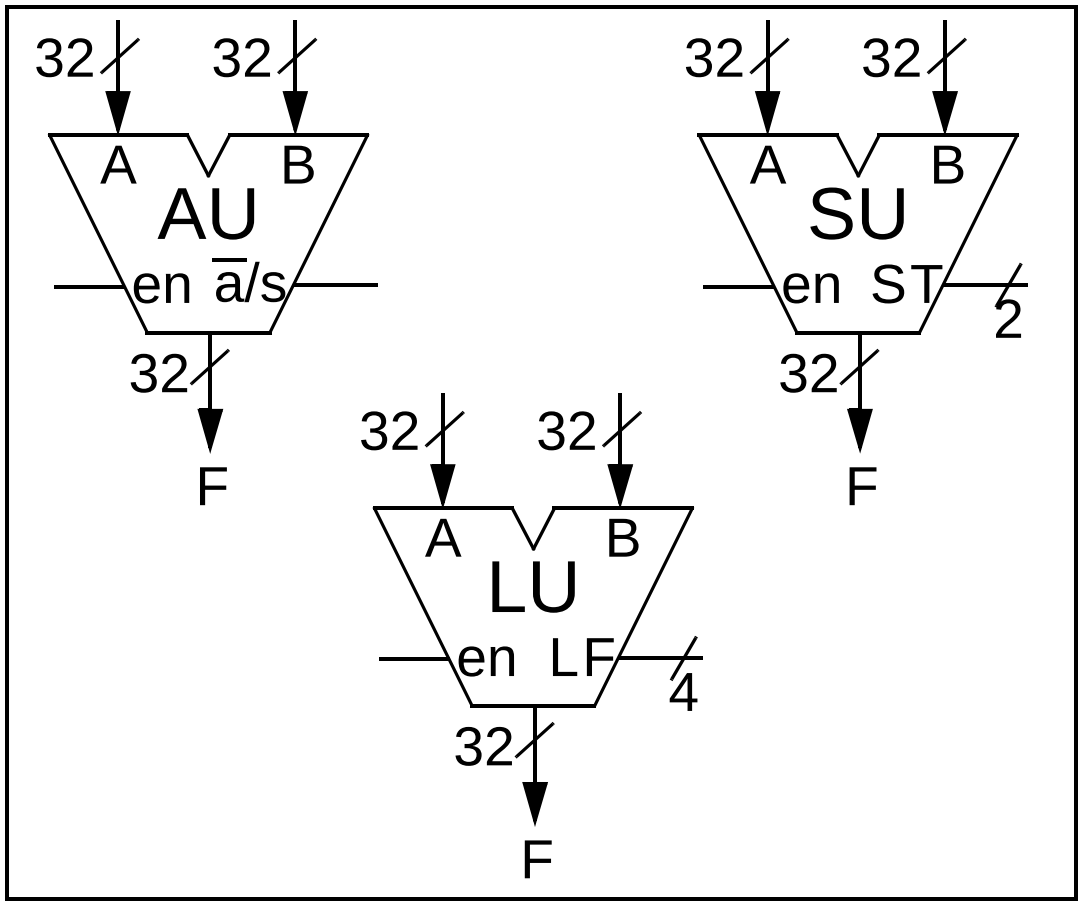


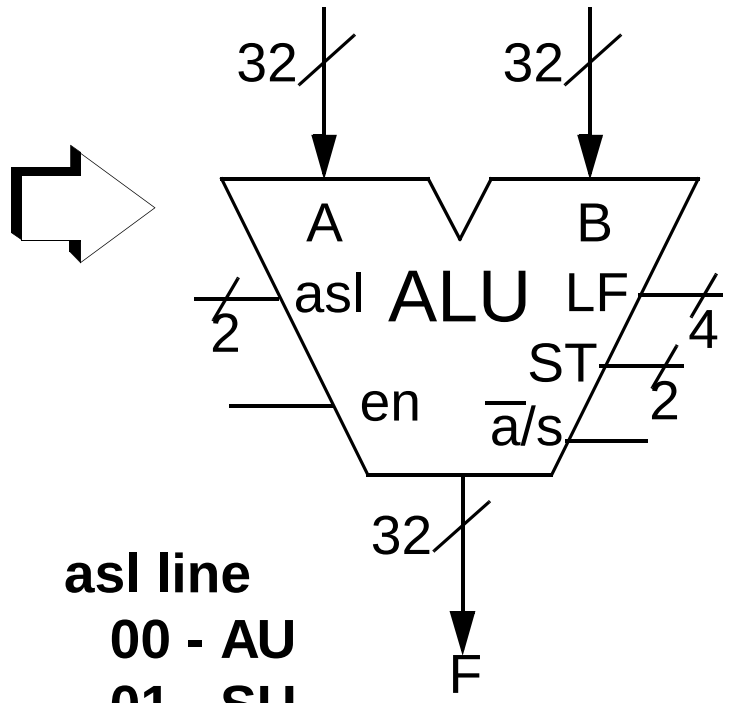
- Notice that the *n*-bit value **A** will be shifted according to the distance indicated with signed number **B**.

SINGLE CYCLE DPU

ARITHMETIC LOGIC UNIT

- 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).



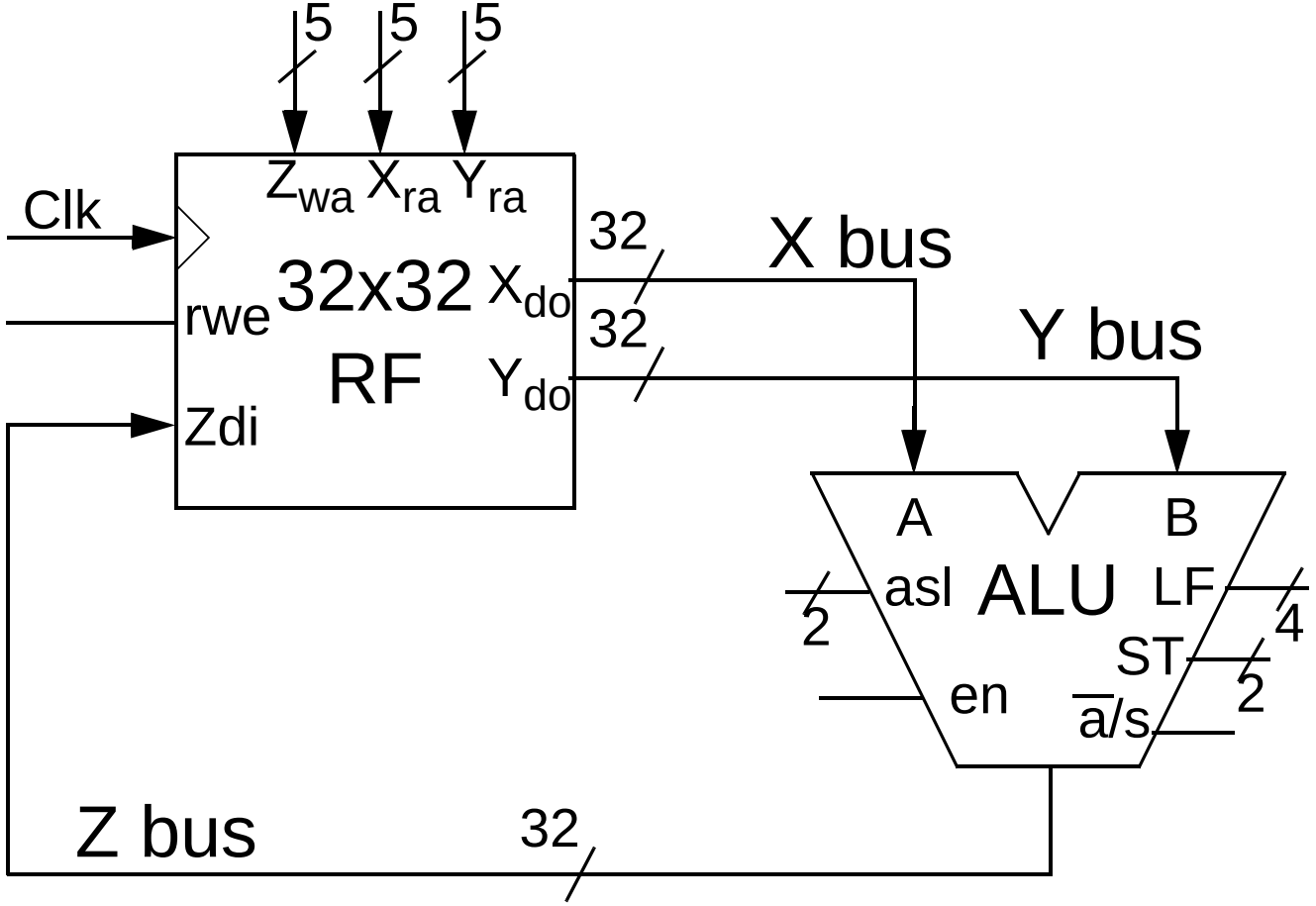


asl line
00 - AU
01 - SU
10 - LU

SINGLE CYCLE DPU

SINGLE CYCLE DPU W/ALU

- Using our **ALU**, the DPU can be redrawn as follows.



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

SINGLE CYCLE DPU

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.

SINGLE CYCLE DPU

MICROCODE

- 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

$$\mathbf{R3} = \mathbf{R1} + \mathbf{R2}$$

or

$$\mathbf{M}[\mathbf{R5}] = \mathbf{R9}$$

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

SINGLE CYCLE DPU

READING FROM MEMORY

- We wish to be able to read and write from our memory.
- A sample read/load operation can be expressed as follows

$$\mathbf{R4} = \mathbf{M}[\mathbf{R7}]$$

  - 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**.
    - $\mathbf{X_{ra}}$ **= 00111**, $\mathbf{Y_{ra}}$ **= XXXXX**, $\mathbf{Z_{wa}}$ **= 00100**, and **rwe = 1** for **RF**.
    - **st_en = 0** and **ld_en = 1**
    - **~r/w = r** and **msel = 1**

SINGLE CYCLE DPU

WRITING TO MEMORY

- A sample write/store operation can be expressed as follows

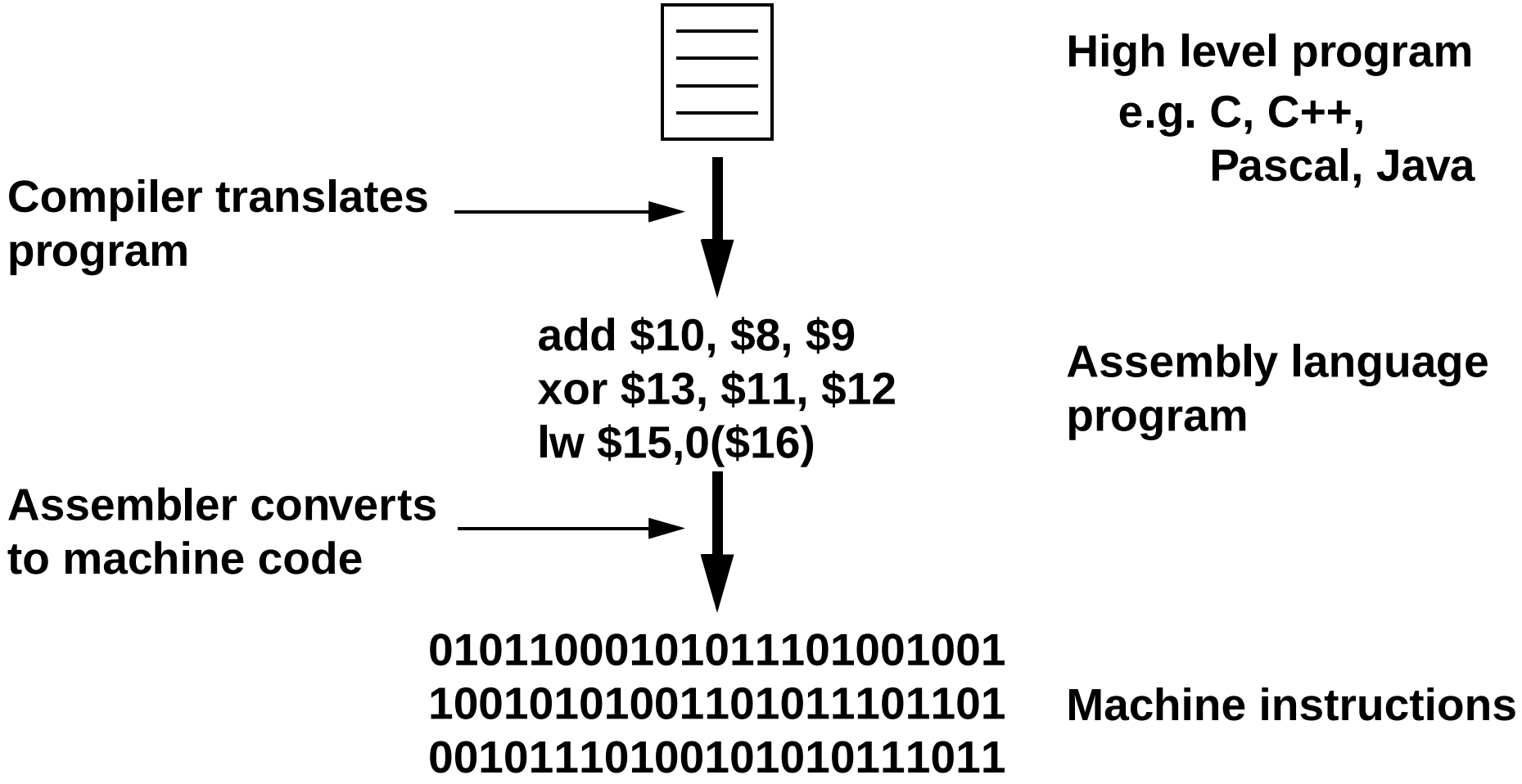
M[R5] = R9

  - 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**.
    - $\mathbf{X_{ra} = 00101}$, $\mathbf{Y_{ra} = 01001}$, $\mathbf{Z_{wa} = XXXXX}$, and **rwe = 0** for **RF**.
    - **st_en = 1** and **ld_en = 0**
    - **~r/w = w** and **msel = 1**

PROGRAM PATH

TRANSLATING CODE

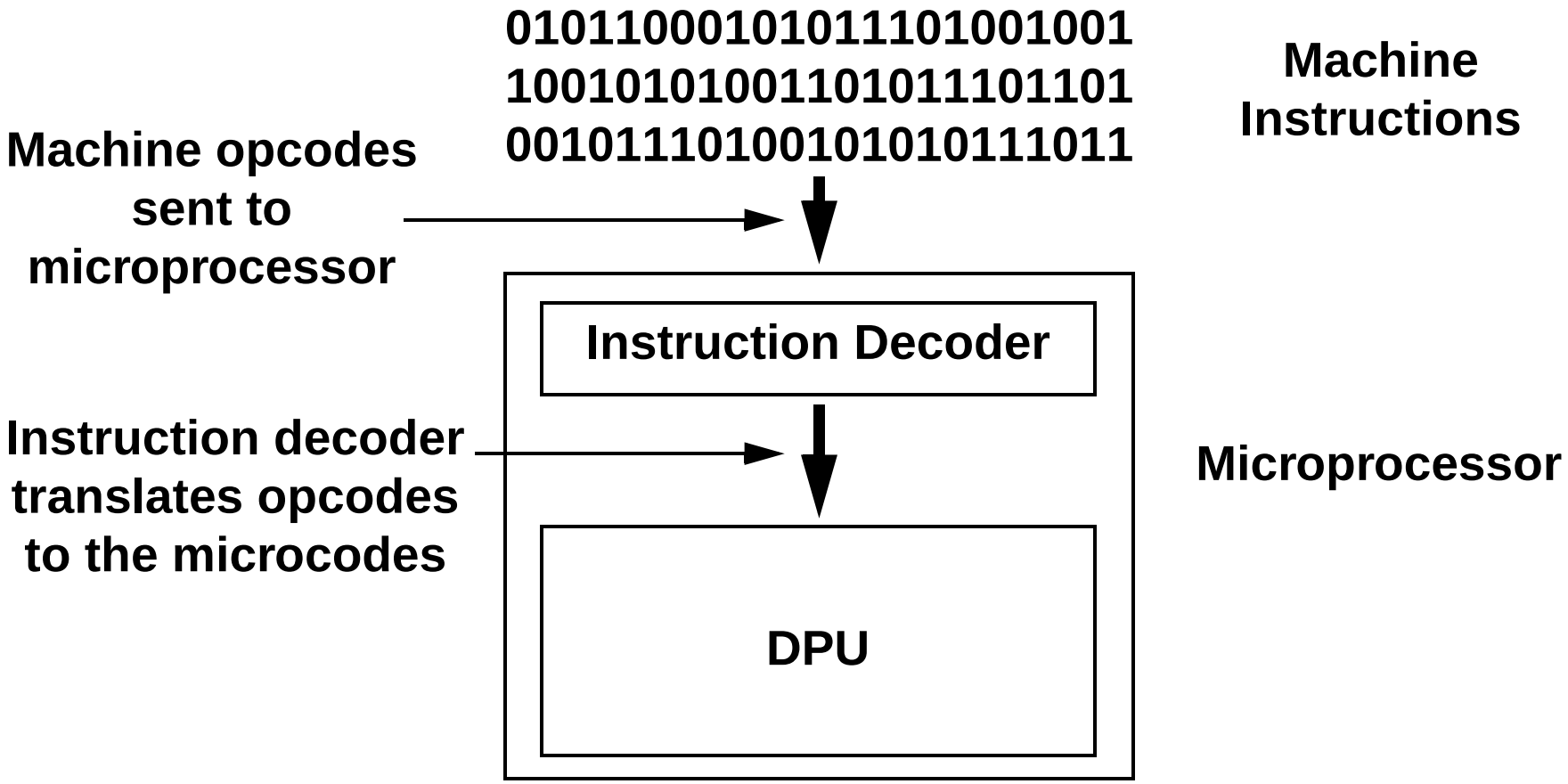
- Below is the process for translating a program to machine opcodes.



PROGRAM PATH

EXECUTING CODE

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

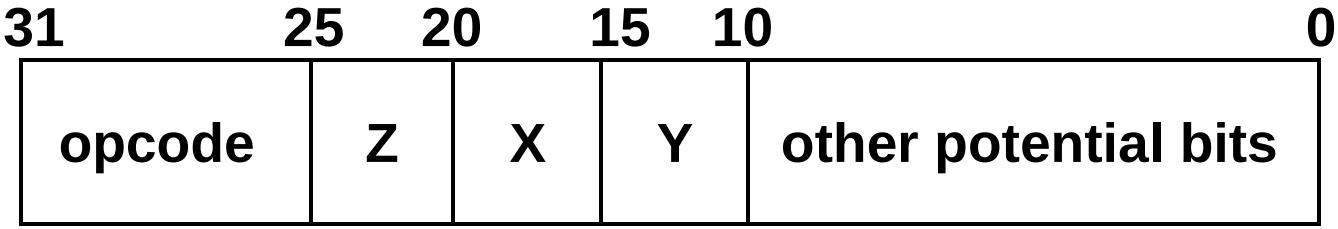


INSTRUCTIONS

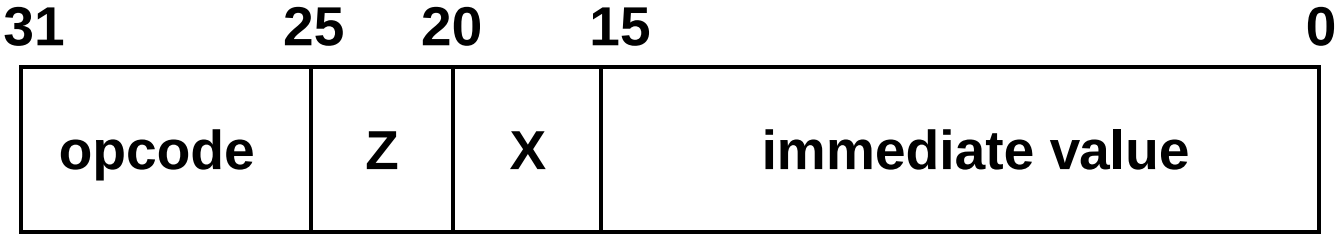
INSTRUCTION FORMATS

- 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



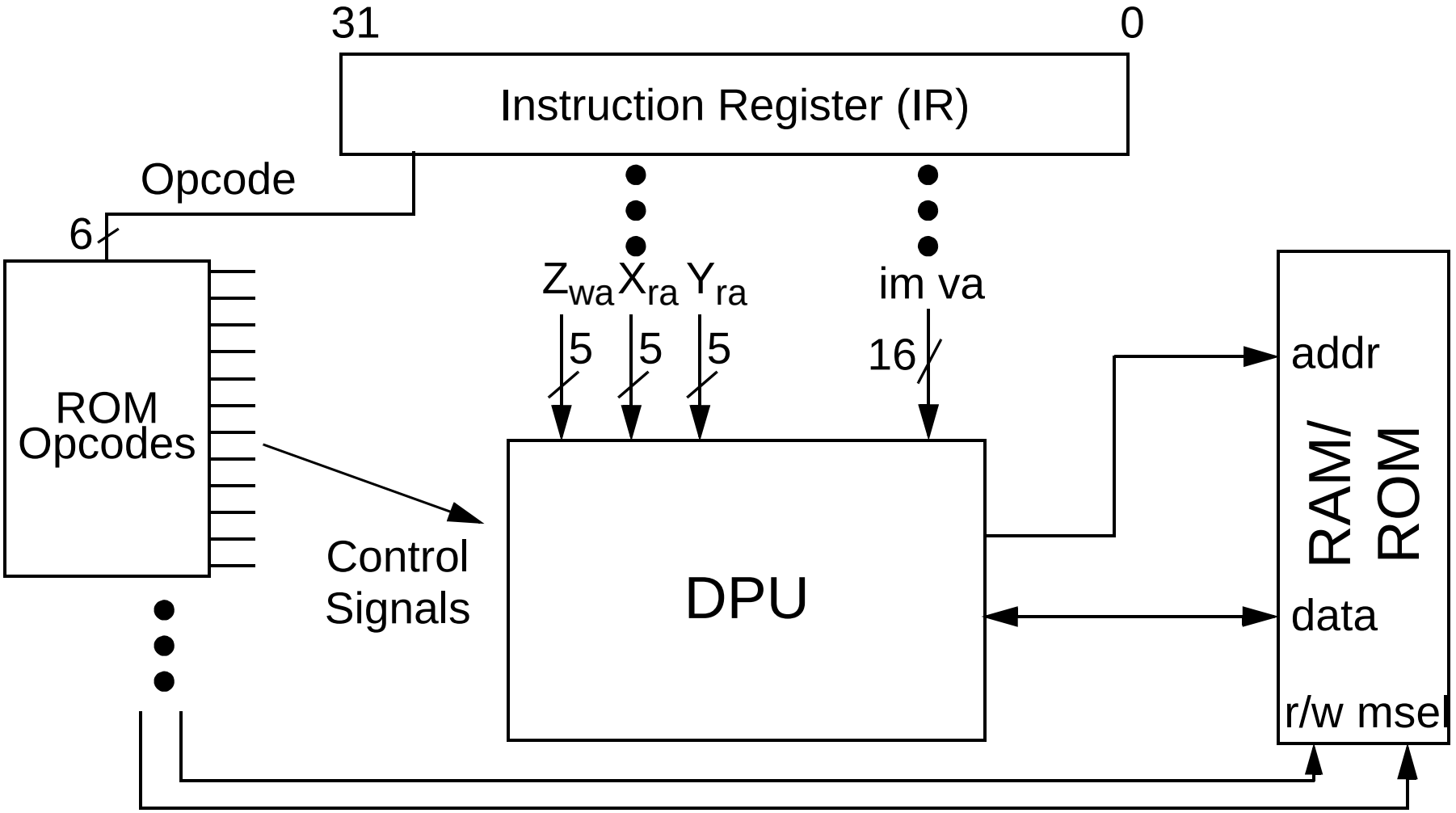
I-format



INSTRUCTIONS

INSTRUCTION REGISTER

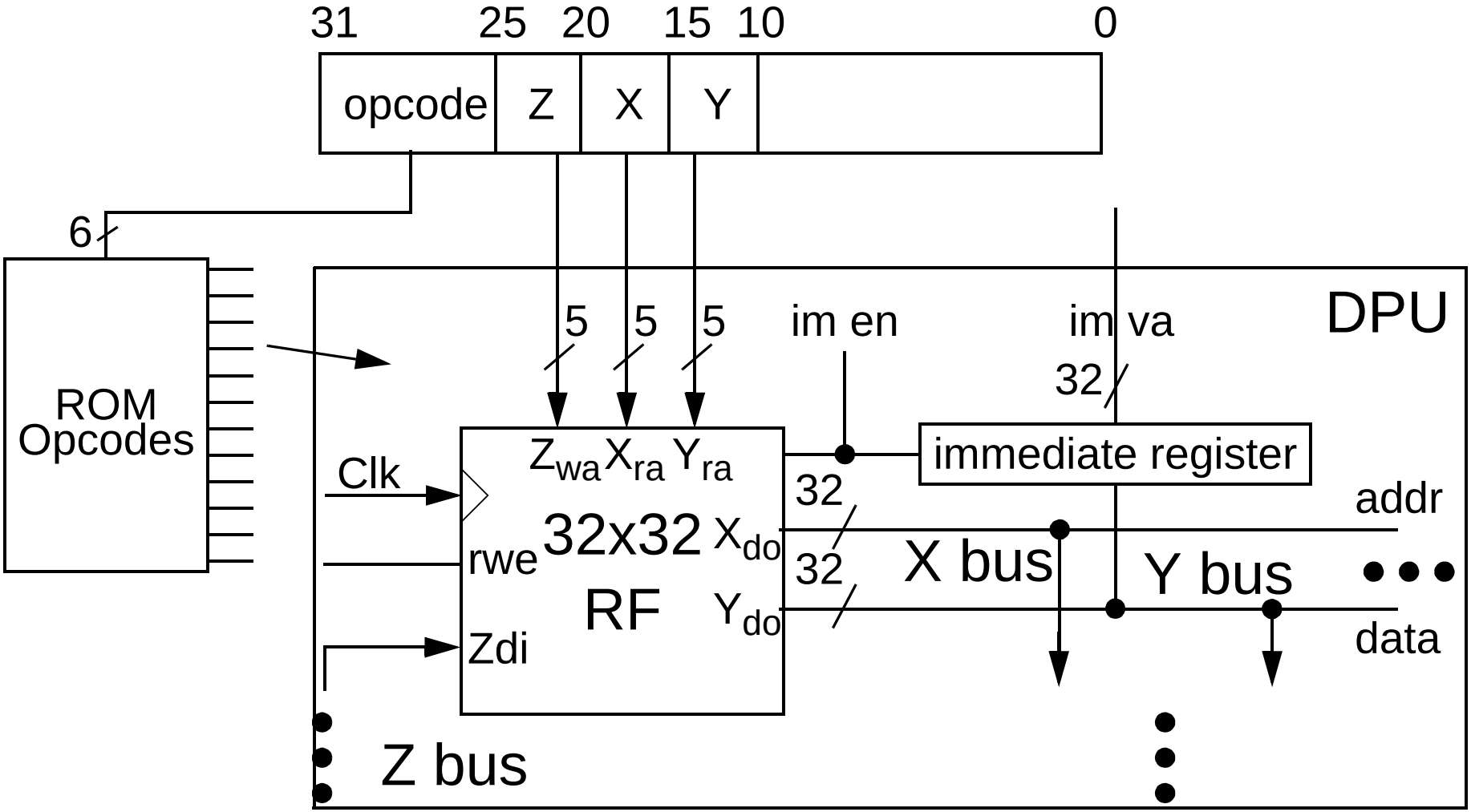
- Use a general instruction register that can act as R- or I-Format.



INSTRUCTIONS

R-FORMAT W/ DPU

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



INSTRUCTIONS

I-FORMAT W/ DPU

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