CHAPTER XII

SINGLE CYCLE DATAPATH UNIT

READ SINGLE CYCLE DATAPATH FREE-DOC ON COURSE WEBPAGE

SINGLE CYCLE DPU

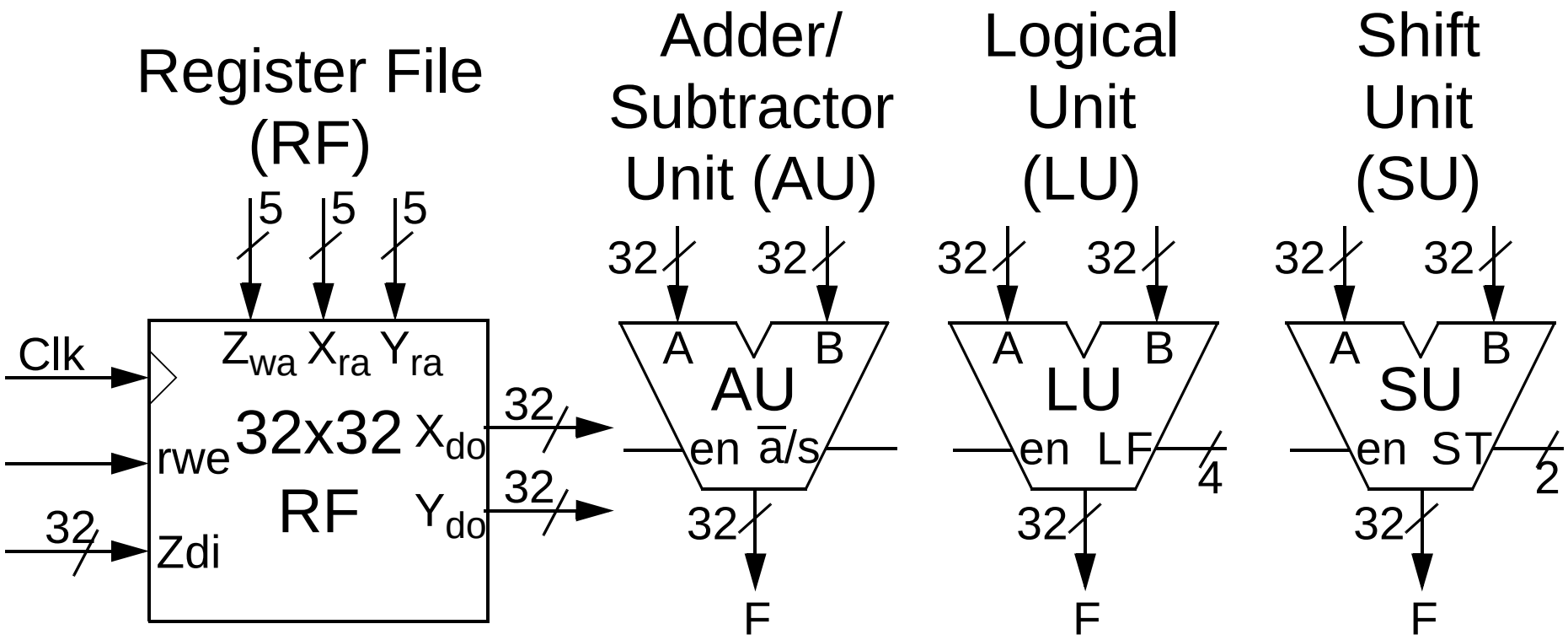
INTRODUCTION

- From the previous chapter, we now have a number of datapath elements such as
  - Register file (**RF**)
  - Adder/subtractor unit (**AU**)
  - Logical unit (**LU**)
  - Shift unit (**SU**)
- The question now is how to take these datapath elements and form a **datapath unit** (DPU).
- The DPU that we will focus on in this chapter is a basic single cycle DPU using a *triple bus internal architecture*.

SINGLE CYCLE DPU

DATAPATH ELEMENTS

- For our examples, we will use the following 32-bit type DPU elements.

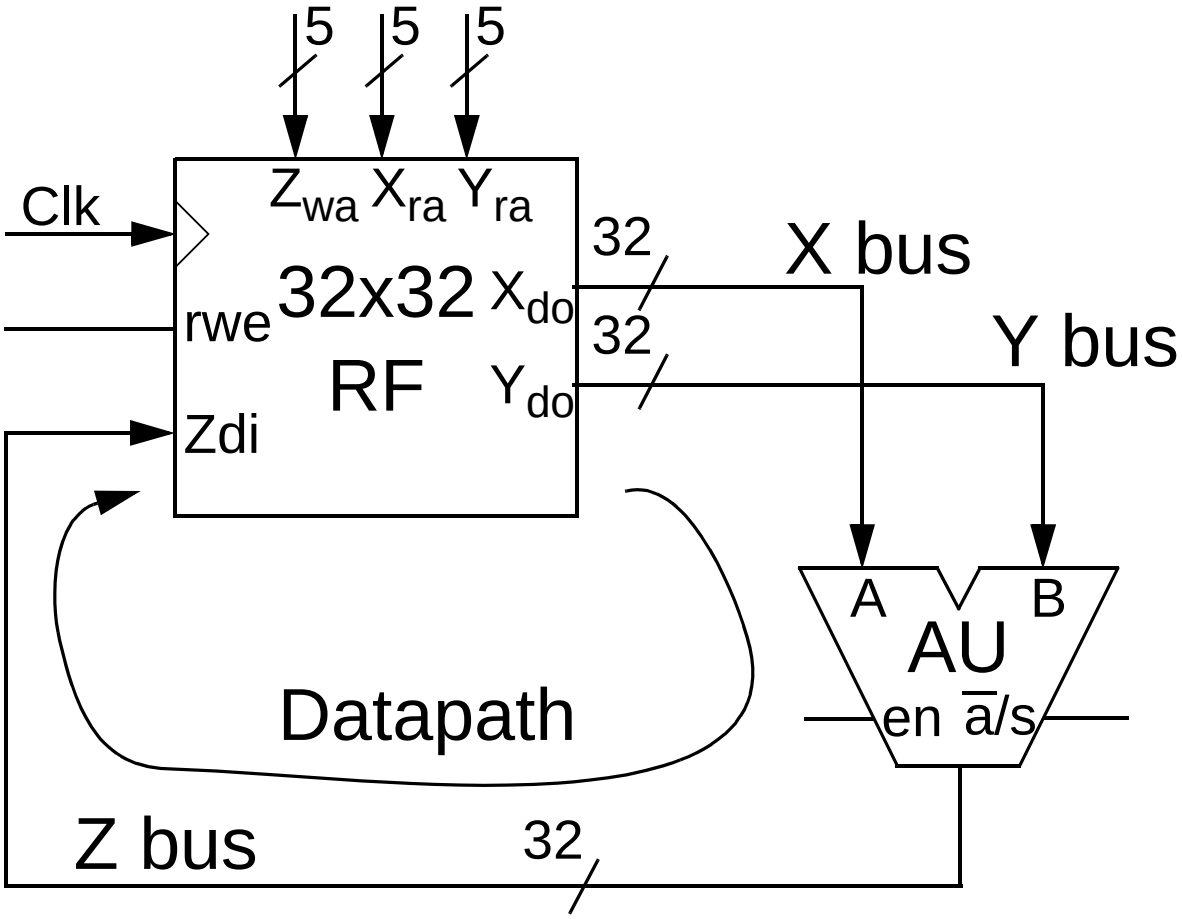


- These allow us to design a 32-bit word computer with 32 registers.
- Of course, other word sizes could be used for other designs.

SINGLE CYCLE DPU

ADD/SUBTRACT MACHINE

- Below is a simple datapath with a register file and adder/subtractor.



Important:
It only takes 1 clock cycle to add/subtract and store the result.

- This structure is also known as a **triple bus internal DPU architecture**.

SINGLE CYCLE CPU

ADD/SUBTRACT MACHINE

- This simple add/subtract machine DPU allows us to add or subtract values in our registers and store the result back into another register.
  - For instance, say that we wanted to **add** the contents of register **R1** with register **R2** and store the result back in register **R3**.

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

  - What control signals are required?
    - $\bar{\mathbf{a}}/\mathbf{s} = \mathbf{0}$ and **en = 1** for **AU**.
    - $\mathbf{X_{ra}} = \mathbf{00001}$, $\mathbf{Y_{ra}} = \mathbf{00010}$, $\mathbf{Z_{wa}} = \mathbf{00011}$, and **rwe = 1** for **RF**.
  - These control signals are applied at the beginning of a clock cycle. The signals then propagate forming the sum at the output of the **AU**. At the end of the clock cycle, the sum (on **Z bus**) is clocked into **R3**.

SINGLE CYCLE CPU

ADD/SUBTRACT MACHINE

- What if instead we wanted to perform the following operation.

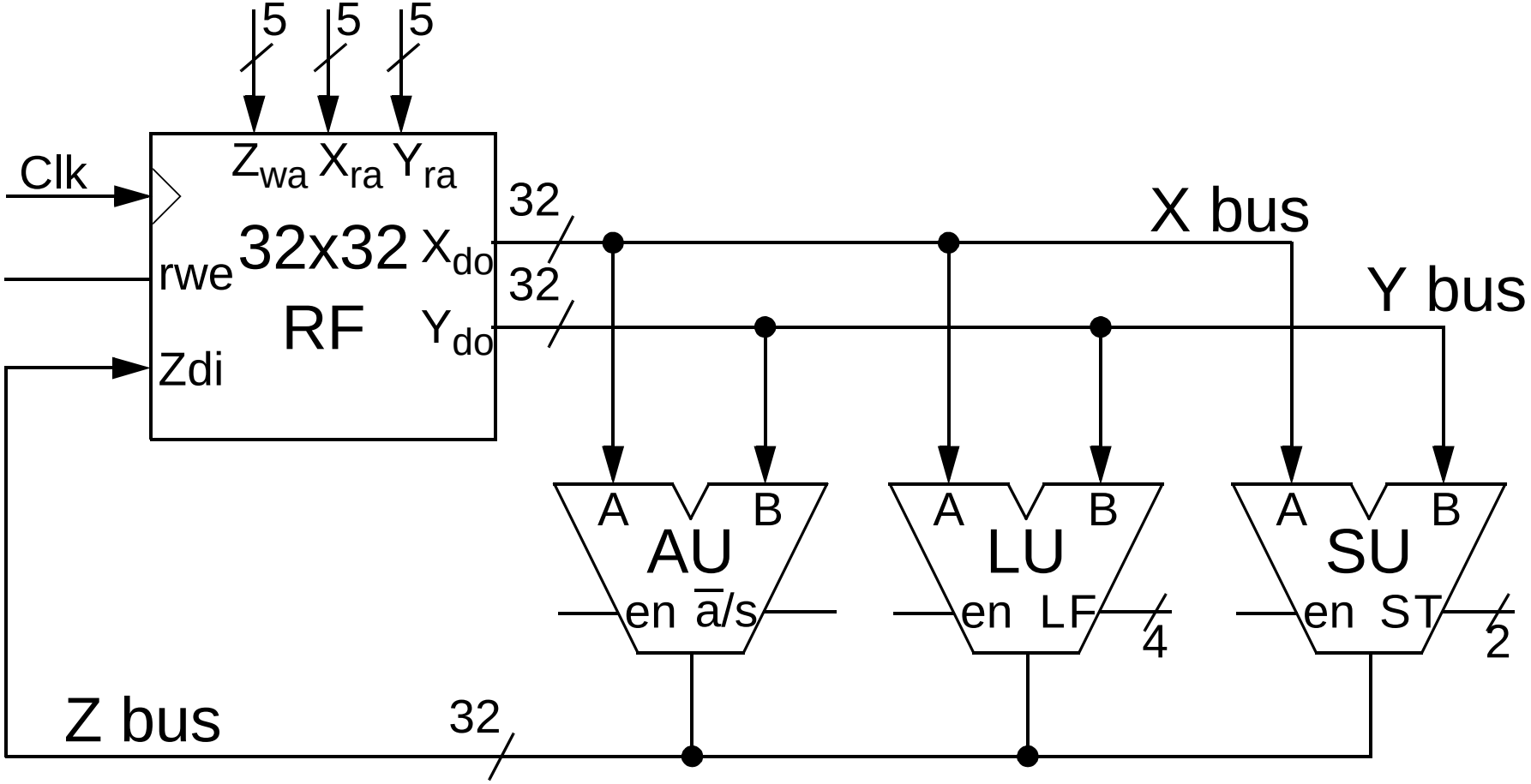
$$\mathbf{R2} = \mathbf{R1} + \mathbf{R2}$$

  - The control signals required are
    - $\bar{\mathbf{a}}/\mathbf{s} = \mathbf{0}$ and **en = 1** for **AU**.
    - $\mathbf{X_{ra}} = \mathbf{00001}$, $\mathbf{Y_{ra}} = \mathbf{00010}$, $\mathbf{Z_{wa}} = \mathbf{00010}$, and **rwe = 1** for **RF**.
  - What is the result of this if the current value of **R1=0x00000001** and **R2=0x00000003**?
    - The register **R2** would be updated at the end of the clock cycle with the value **0x00000004**.
  - Remember, the current value of **R2** is put on the **X** or **Y bus**, and it is only at the **END** of the clock cycle that the contents of **R2** get changed.

SINGLE CYCLE DPU

BASIC SINGLE CYCLE DPU

- A more useful single cycle datapath can be as follows.



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

SINGLE CYCLE CPU

COMPUTATION EXAMPLES

- How does this change the additions we were doign earlier?
  - Say we want to again perform the following addition.

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

  - The control signals we would need are
    - $\bar{\mathbf{a}}/\mathbf{s} = \mathbf{0}$ and **en = 1** for **AU**.
    - **en = 0** for **LU**.
    - **en = 0** for **SU**.
    - $\mathbf{X_{ra} = 00001}$, $\mathbf{Y_{ra} = 00010}$, $\mathbf{Z_{wa} = 00011}$, and **rwe = 1** for **RF**.
  - Notice that we use the same control signals as before, but now include signals to disable the **LU** and **SU** during this addition clock cycle.

SINGLE CYCLE DPU

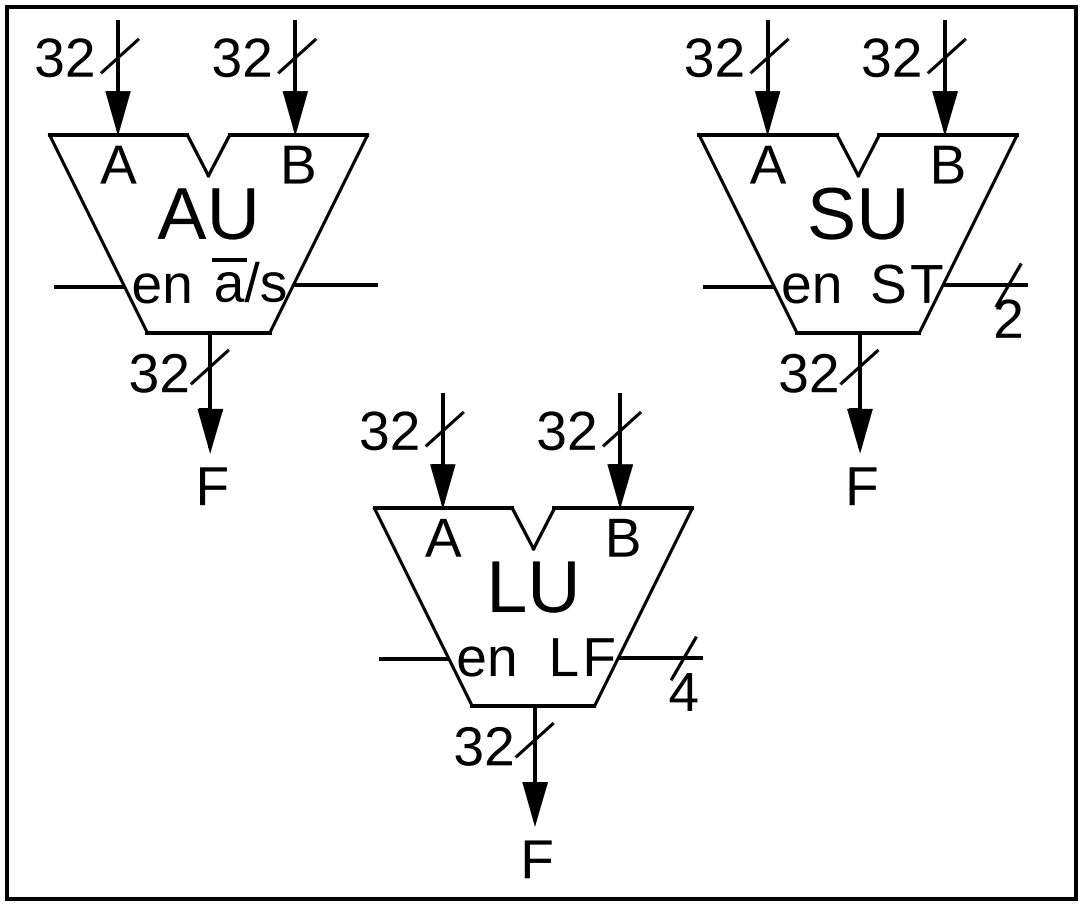
COMPUTATION EXAMPLES

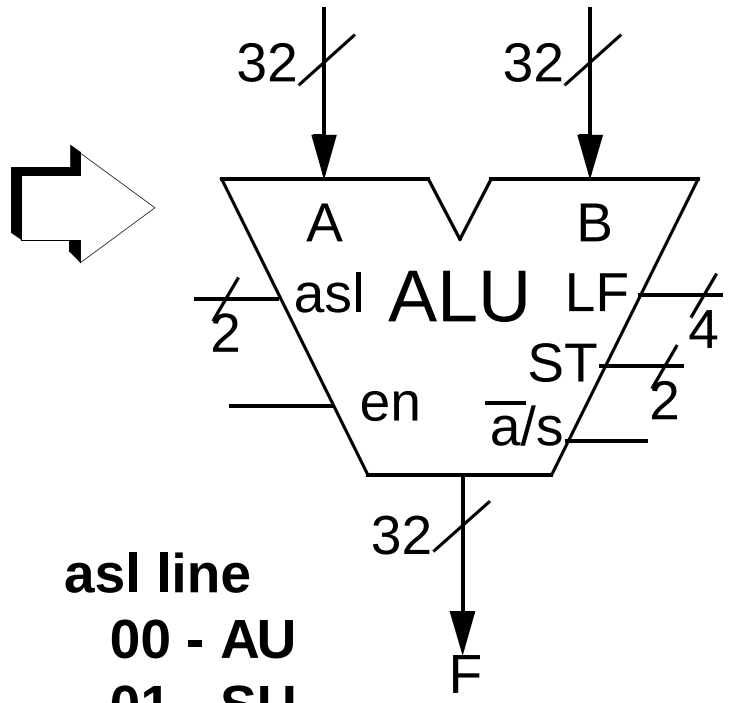
- Another operation we might want to do with this DPU is perform a logical shift of the contents of **R15** by a distance indicated in **R6**.
  - The control signals required are
    - **en = 0** for **AU**.
    - **en = 0** for **LU**.
    - **en = 1** and **ST = 00** for **SU**.
    - $\mathbf{X_{ra} = 01111}$, $\mathbf{Y_{ra} = 00110}$, $\mathbf{Z_{wa} = 01111}$, and **rwe = 1** for **RF**.
  - Notice that this set of control signals disables the **AU** and **LU** while enabling the **SU**.
  - The **SU** is set to do a logical shift with **ST = 00**.
  - The distance of the shift is according to what is in **R6**.
  - The result is stored back in **R15** with $\mathbf{Z_{wa} = 01111}$ and **rwe = 1** for **RF**.

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).



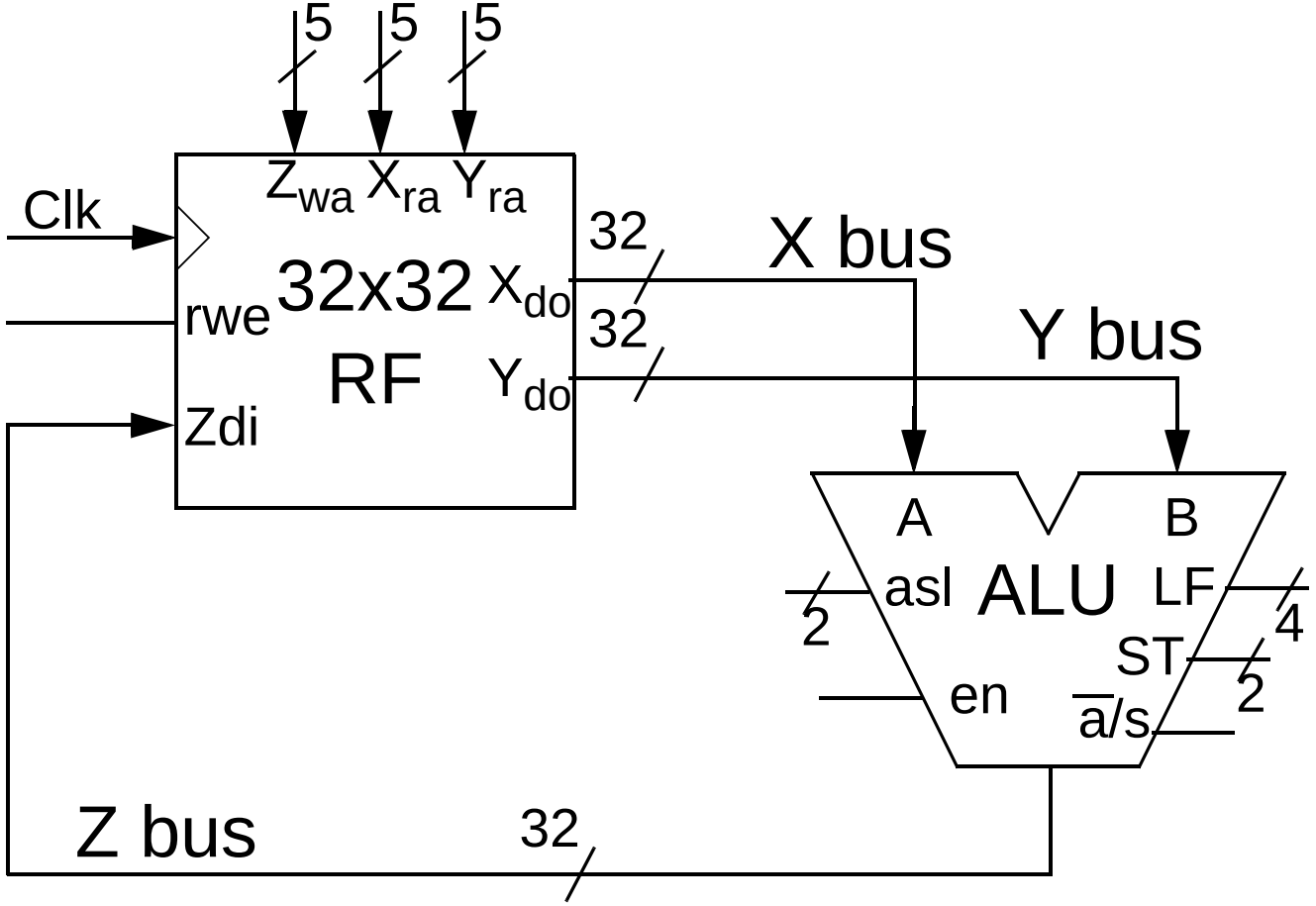


asl line
00 - AU
01 - SU
10 - LU

SINGLE CYCLE DPU

SINGLE CYCLE DPU W/ALU

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

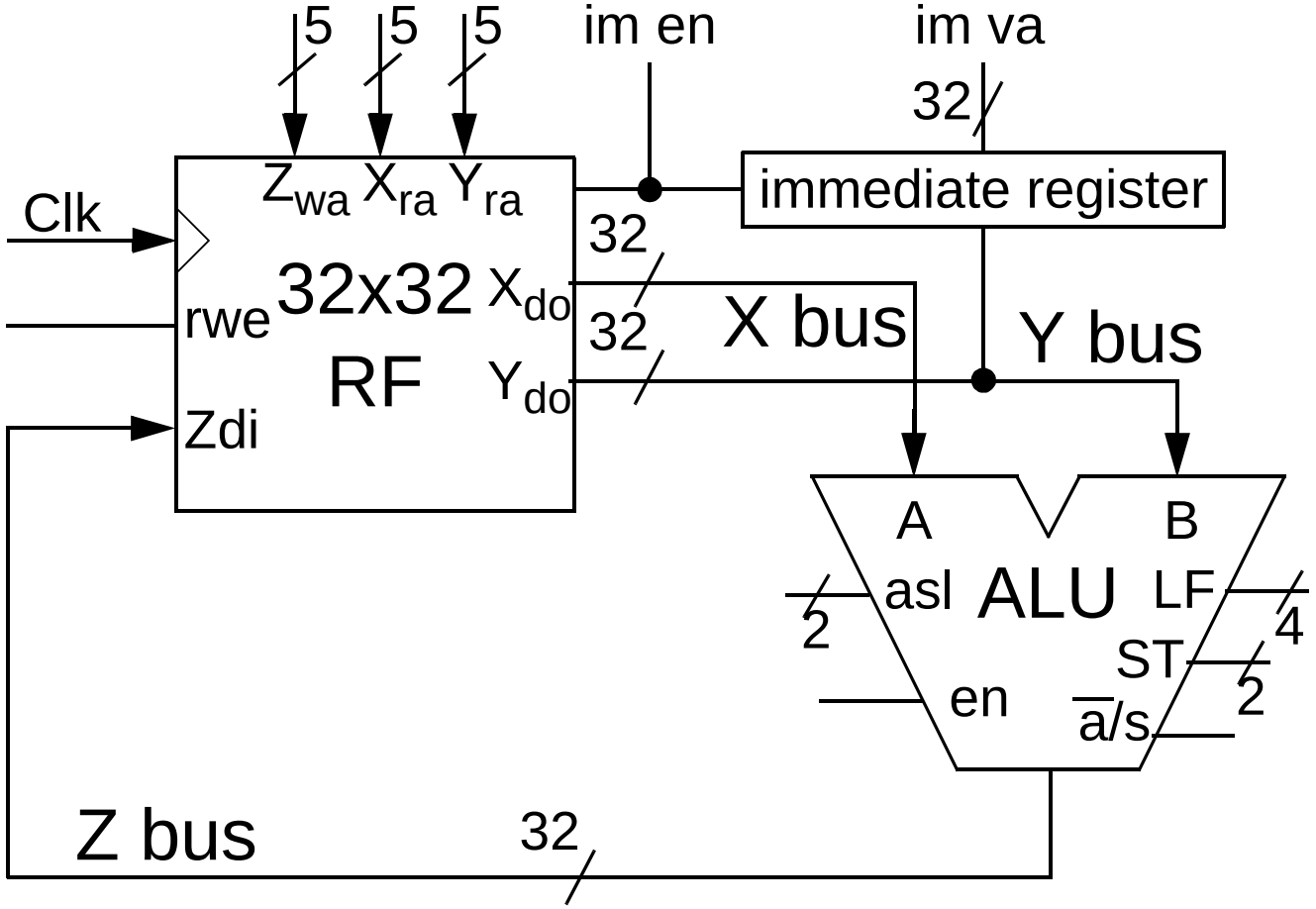


- This structure is still a **triple bus internal DPU architecture**.

SINGLE CYCLE DPU

IMMEDIATE REGISTER

- Many designs also include some form of immediate register.



- Allows for operations such as **R28** = **R5** + **Immediate**.

SINGLE CYCLE DPU

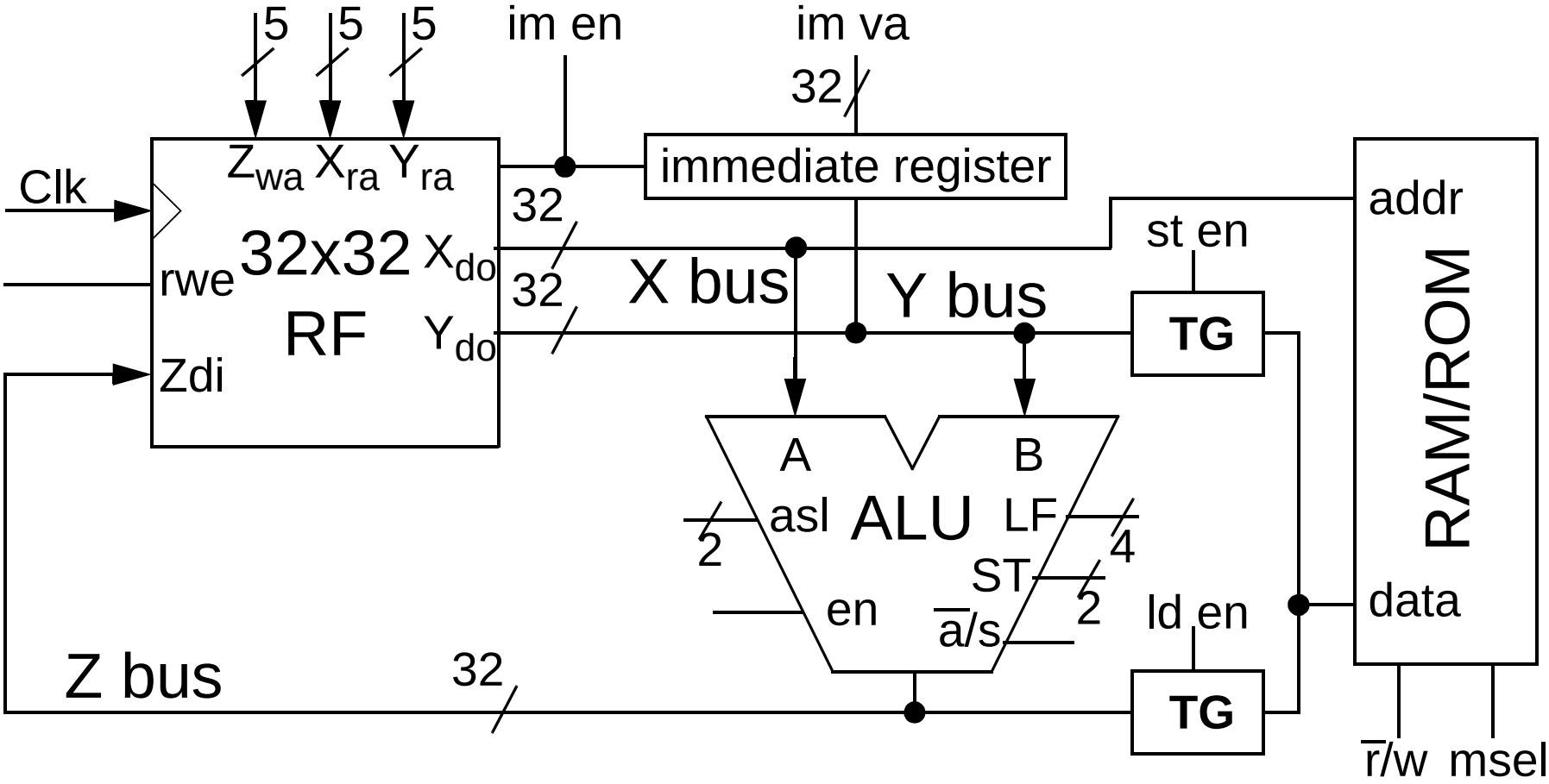
IMMEDIATE REGISTER

- 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

INCLUDING MEMORY

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



SINGLE CYCLE DPU

INCLUDING MEMORY

- The included has the following characteristics
  - 32 address lines
  - 32 data lines
  - a read/write line
  - a chip select or memery select line
- Two transmission gates block the bidirectional data lines for the memory.
  - Notice that if **st_en** is high, then we can potentially write to the memory.
  - Notice that if **ld_en** is high, then we can potentially read from the memory.

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**

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.