

8- DATAPATH

As mentioned, the CPU can be divided into a **data section** and a **control section**. The **data section**, which is also called the datapath, contains the registers and the ALU. The datapath is capable of performing certain operations on data items.

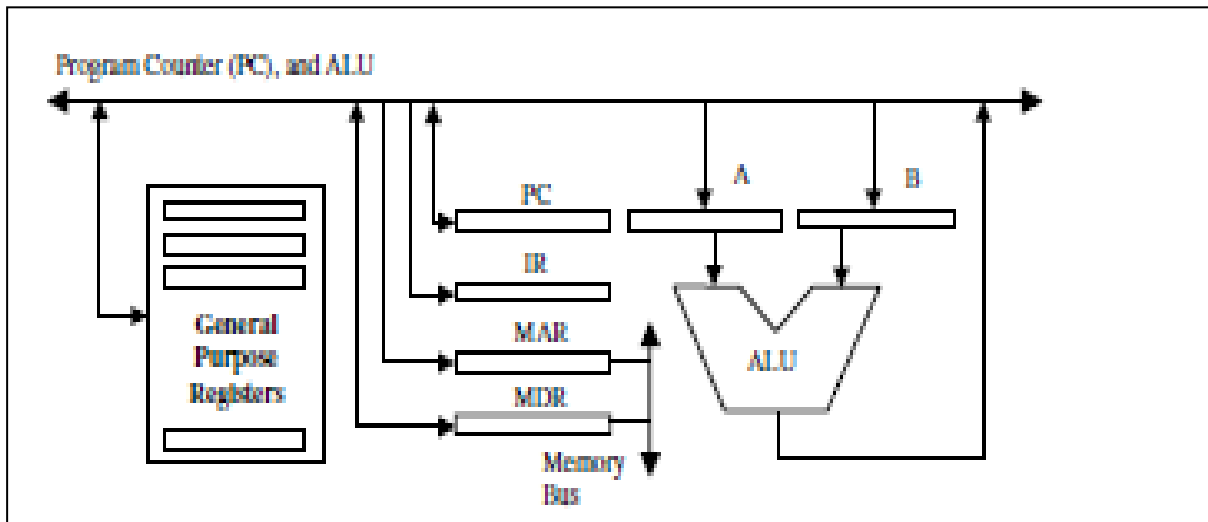
The **control section** is basically the control unit, which issues control signals to the datapath. Internal to the CPU, data move from one register to another and between ALU and registers. Internal data movements are performed via **local buses, which may carry data, instructions, and addresses**. Externally, data move from registers to memory and I/O devices, often by means of a system bus. Internal data movement among registers and between the ALU and registers may be carried out using different organizations including **one-bus, two-bus, or three-bus organizations**. Dedicated datapaths may also be used between components that transfer data between themselves more frequently. For example, the contents of the PC are transferred to the MAR to fetch a new instruction at the beginning of each instruction cycle. Hence, a dedicated datapath from the PC to the MAR could be useful in speeding up this part of instruction execution.

8-1 One-Bus Organization

Using one bus, the CPU registers and the ALU use a single bus to move outgoing and incoming data. Since a bus can handle **only a single data movement within one clock cycle, two-operand operations will need two cycles to fetch the operands for the ALU**. Additional registers may also be needed to buffer data for the ALU.

This bus organization is the **simplest and least expensive**, but it **limits the amount of data transfer that can be done in the same clock cycle, which will slow down the overall performance**. Figure (8.1) shows a one-bus datapath consisting of a set of general-purpose registers, a memory address register (MAR),

a memory data register (MDR), an instruction register (IR), a program counter (PC), and an ALU.

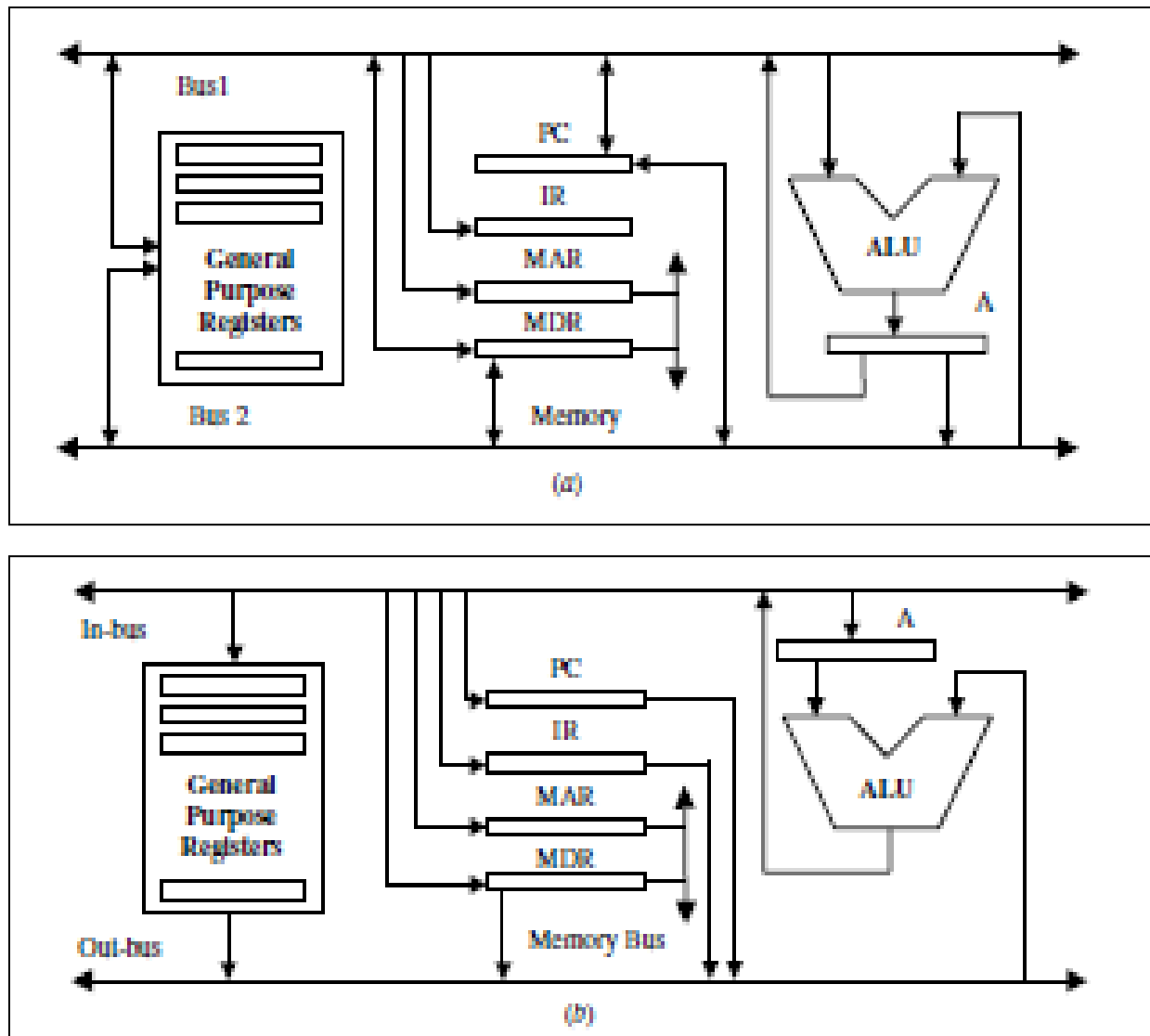


Figure(8.1) One- Bus datapath

8-2 Two-Bus Organization

Using two buses is a **faster solution** than the one-bus organization. In this case, **general-purpose registers are connected to both buses**. Data can be transferred from two **different registers to the input point of the ALU at the same time**. Therefore, a two operand operation can fetch both operands in the same clock cycle. An additional buffer register may be needed to hold the output of the ALU when the two buses are busy carrying the two operands. Figure (8.2)(a) shows a two-bus organization.

In some cases, one of the buses may be dedicated for moving data into registers (in-bus), while the other is dedicated for transferring data out of the registers (out-bus). In this case, the additional buffer register may be used, as one of the ALU inputs, to hold one of the operands. The ALU output can be connected directly to the in-bus, which will transfer the result into one of the registers. (8.2)(b) shows a two-bus organization with in-bus and out-bus.

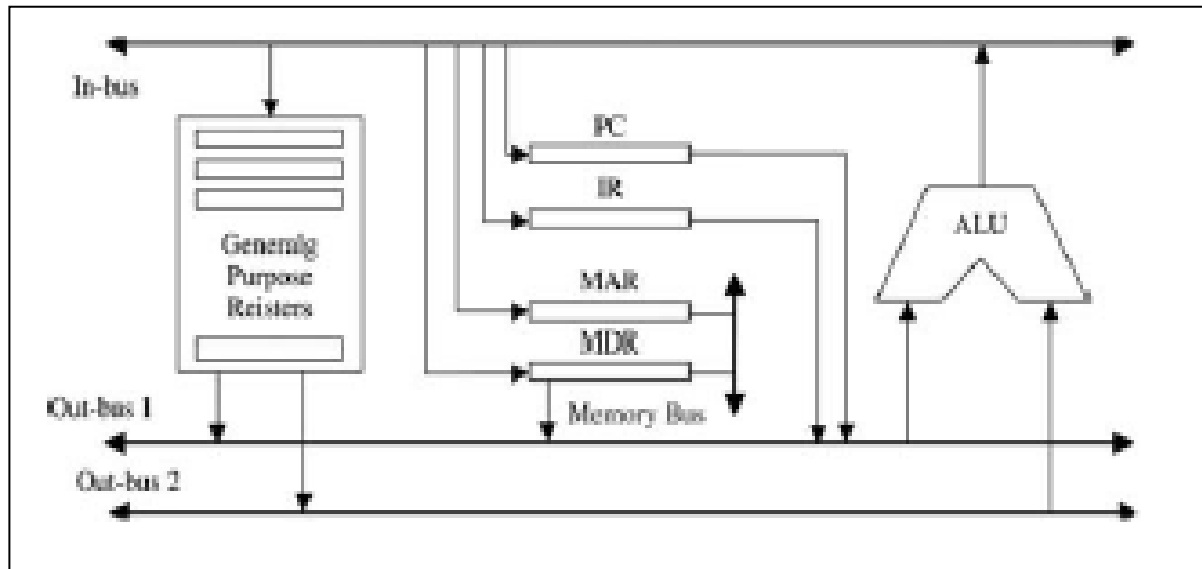


Figure(8.2) Two – Bus Organizations. (a) An example of Two-Bus Datapath. (b) Another example of Two-Bus Datapath with in-Bus and Out-bus.

8-3 Three-Bus Organization

In a three-bus organization, two buses may be used as source buses while the third is used as destination. The source buses move data out of registers (out-bus), and the destination bus may move data into a register (in-bus). Each of the two out-buses is connected to an ALU input point. The output of the ALU is connected directly to the in-bus. As can be expected, the more buses we have, the more data

we can move within a single clock cycle. However, increasing the number of buses will also increase the complexity of the hardware. Figure (8.3) shows an example of a three-bus datapath.



Figure(8.3) Tree- Bus datapath

8- 4 Examples : Generate the **control sequence** of the execution for the following instructions using **One(single),Two and Three bus organizations** .

1- Add R1,R2

One-Bus :

- 1- Pc_{out} , AR_{in} , Read , Set carry of ALU, Clear y, Add, Z_{in} .
- 2- Z_{out} , Pc_{in} , WMFC (Wait for Memory-Function-Completed signal).
- 3- DR_{out} , IR_{in} .
- 4- $R1_{out}$, y_{in} .
- 5- $R2_{out}$, add , Z_{in}
- 6- Z_{out} , $R1_{in}$
- 7- End.

Two-Bus : (according to figure (8.2)(b))

- 4- $R1_{out}, y_{in}$.
- 5- $R2_{out}, add, R1_{in}$
- 6- End.

Three-Bus :

- 4- $R1_{out}, R2_{out}, add, R1_{in}$
- 5- End.

2- Add $R1, [R3]$

One-Bus :

- 1- $Pc_{out}, AR_{in}, Read, Set\ carry\ of\ ALU, Clear\ y, Add, Z_{in}$.
- 2- $Z_{out}, Pc_{in}, WMFC$ (Wait for Memory-Function-Completed signal).
- 3- DR_{out}, IR_{in} .
- 4- $R3_{out}, AR_{in}, Read$.
- 5- $R1_{out}, y_{in}, WMFC$
- 6- DR_{out}, add, Z_{in}
- 7- $Z_{out}, R1_{in}$
- 8- End.

Two-Bus : (according to figure (8.2)(b))

- 4- $R3_{out}, AR_{in}, Read$.
- 5- $R1_{out}, y_{in}, WMFC$
- 6- $DR_{out}, add, R1_{in}$
- 7- End.

Three-Bus :

- 4- $R3_{out}, AR_{in}, Read, WMFC$.
- 5- $R1_{out}, DR_{out}, add, R1_{in}$
- 6- End.

3- Jmp Z (jump if zero) $\Rightarrow PC = PC + x$ One-Bus :

- 4- If N end
- 5- PC_{out}, y_{in}
- 6- Address of IR_{out} , add , Z_{in}
- 7- Z_{out} , PC_{in}
- 8- End.

Two-Bus : (according to figure (8.2)(b))

- 4- If N end , PC_{out} , y_{in}
- 5- Address of IR_{out} , add , PC_{in}
- 6- End.

Three-Bus :

- 4- PC_{out} , address of IR_{out} , add , if N PC_{in}
- 5- End.