Computer Architecture

Pipeline in CPUs

Dr. Masood Mir

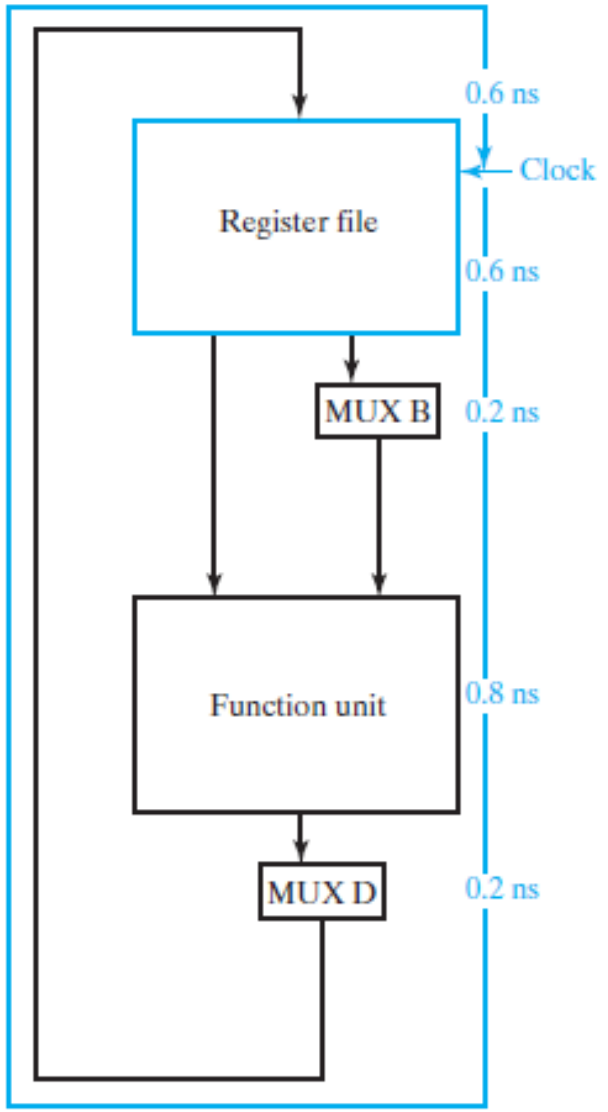
Week 12

RISC and CISC CPUs

- The purpose of main CPU in a computer is to decode instructions received from memory and perform transfer, arithmetic, logic, and control operations with data stored in internal registers, memory, or I/O interface units.
- Other than the main CPU, there may be **more small dedicated CPUs in a computer**. Small, relatively simple CPUs called microcontrollers are used in computers to perform limited or specialized tasks. For example, a microcontroller is present in the keyboard and in the display in the generic computer.
- Thus, different types of CPUs or microcontrollers are required according to their application.
- In such microcontrollers, the CPU may be quite different or simple. The word lengths may be short (e.g., eight bits), the number of registers small, and the instruction sets limited, with low performance.
- To discuss more types of CPUs, first we need to understand "**Pipelined Datapath**". A pipelined datapath is same as datapath discussed previously with the addition of Pipeline. Next, we add a pipelined control unit to form a **reduced instruction set computer (RISC)**. Problems due to the use of pipelining in RISC are introduced and different solutions are discussed.
- Next, the control unit is expanded to form a **complex instruction set computer (CISC)**.
- We will also discuss techniques to enhance pipelined processor performance.

Pipelined Datapath

- Figure 10-1(a) illustrates maximum delay values for each of the components of a typical datapath.
- A maximum of **0.8 ns** (0.6 ns + 0.2 ns) is required **to read two operands** from the register file or to read one operand from the register file and obtain a constant from MUX B.
- A maximum of **0.8 ns** is required to execute an operation in the **functional unit**.
- Also, a maximum of **0.8 ns** is required **to write the result** back into the register file, including the delay of MUX D.
- Adding these delays, we find that **2.4 ns** is required to perform a single microoperation.
- Thus, the maximum rate at which the microoperations can be performed is the inverse of 2.4 ns (i.e., **416.7 MHz**). This is the maximum frequency at which the clock can be operated.



(a) Conventional

Pipelined Datapath

- For the datapath alone and for the combination of the datapath and control unit in the **single-cycle computer**, the execution of a **microoperation** constitutes the **execution of an instruction**.
- Thus, the rate of execution of instructions equals the clock frequency.
- Now suppose that the datapath execution rate is not adequate for a particular application, and that no faster components are available with which to reduce the 2.8 ns required to complete a microoperation. Still, **it may be possible to reduce the clock period and increase the clock frequency**. This can be done by breaking up the 2.8 ns delay path with registers. The resulting datapath, sketched in Figure 10-1(b), is referred to as a pipelined datapath, or just a **pipeline**.

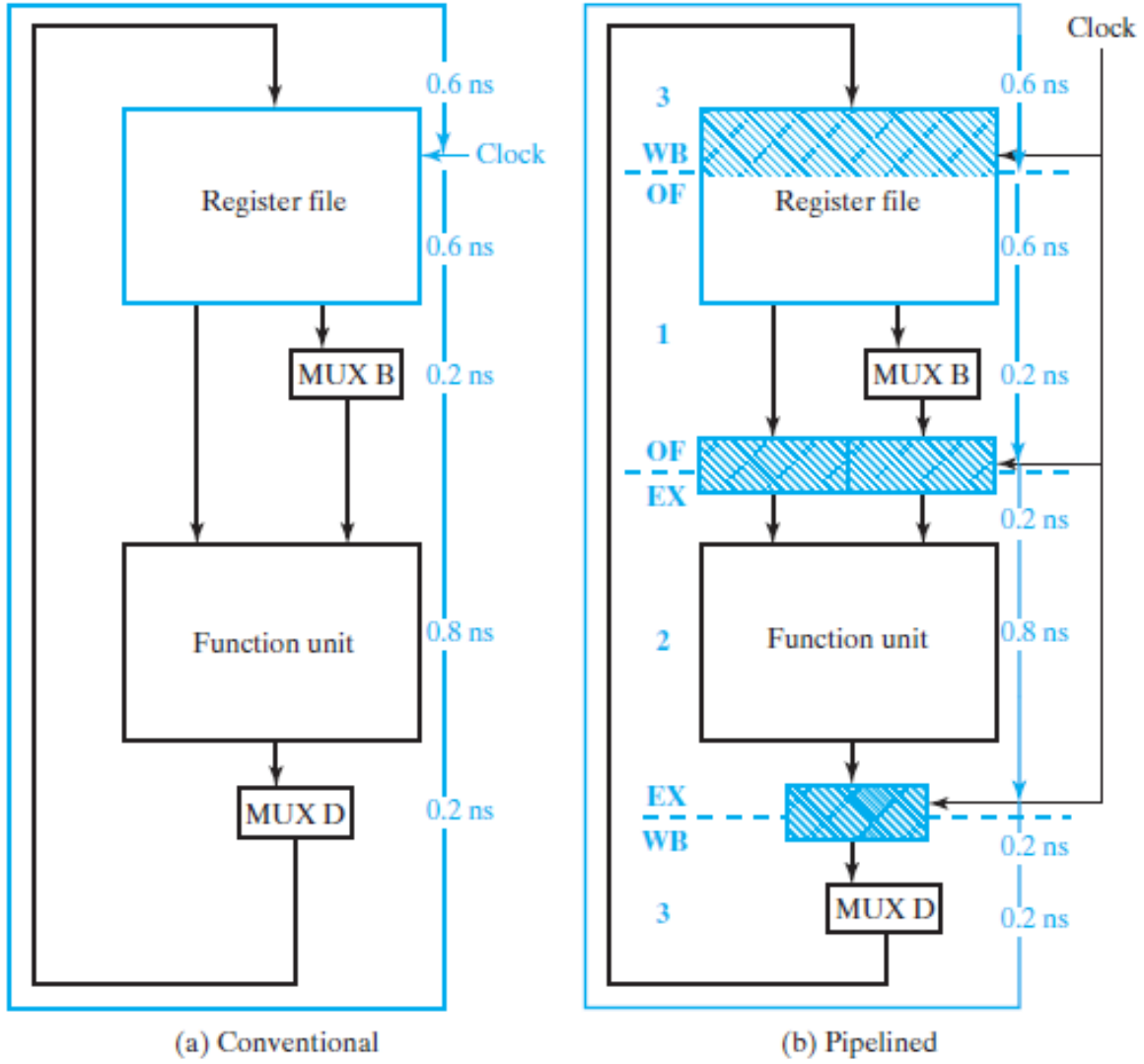


FIGURE 10-1
Datapath Timing

Pipelined Datapath

- There are three main stages:
- **Operand Fetch** (or **OF**)
- **Execution** (or **EX**)
- **Write Back** (or **WB**)
- **Three sets of registers (or platform)** break the original datapath into three parts.
- The two registers that store the A data from the register file and the output of MUX B constitute the one set of registers. (OF stage)
- The second set of registers are after execution in Function Unit to store the inputs to MUX D. (EX stage)
- The register file contains the third set of registers. (WB stage)

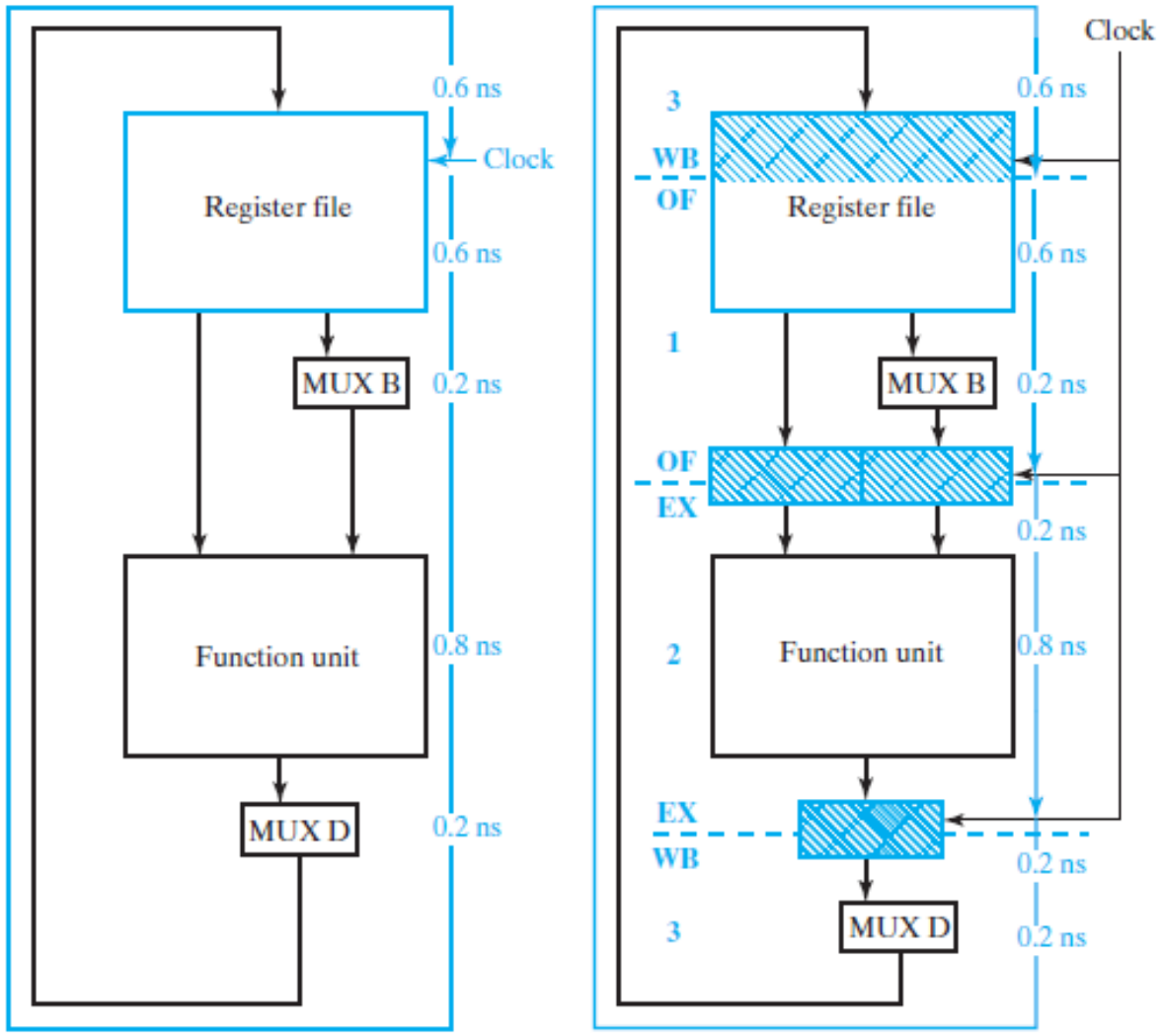


FIGURE 10-1
Datapath Timing

Pipelined Datapath

- Stage 1 of the pipeline has the delay required for reading the register file followed by selection by MUX B. This delay is 0.6 plus 0.2 ns, or 0.8 ns.
- Stage 2 of the pipeline has the 0.2 ns delay of the platform plus the 0.8 ns delay of the functional unit, giving 1.0 ns.
- Stage 3 has the 0.2 ns delay of the platform, the delay for the selection by MUX D, and the delay for writing back into the register file. This delay is 0.2 + 0.2 + 0.6, for a total of 1.0 ns.
- **Thus, all flip-flop–to–flip-flop delays are at most 1.0 ns**.
- If all three stages work in parallel, it allows a minimum clock period of 1.0 ns and a maximum clock frequency of **1.0 GHz**, compared with the 416.7 MHz for the single-stage datapath.

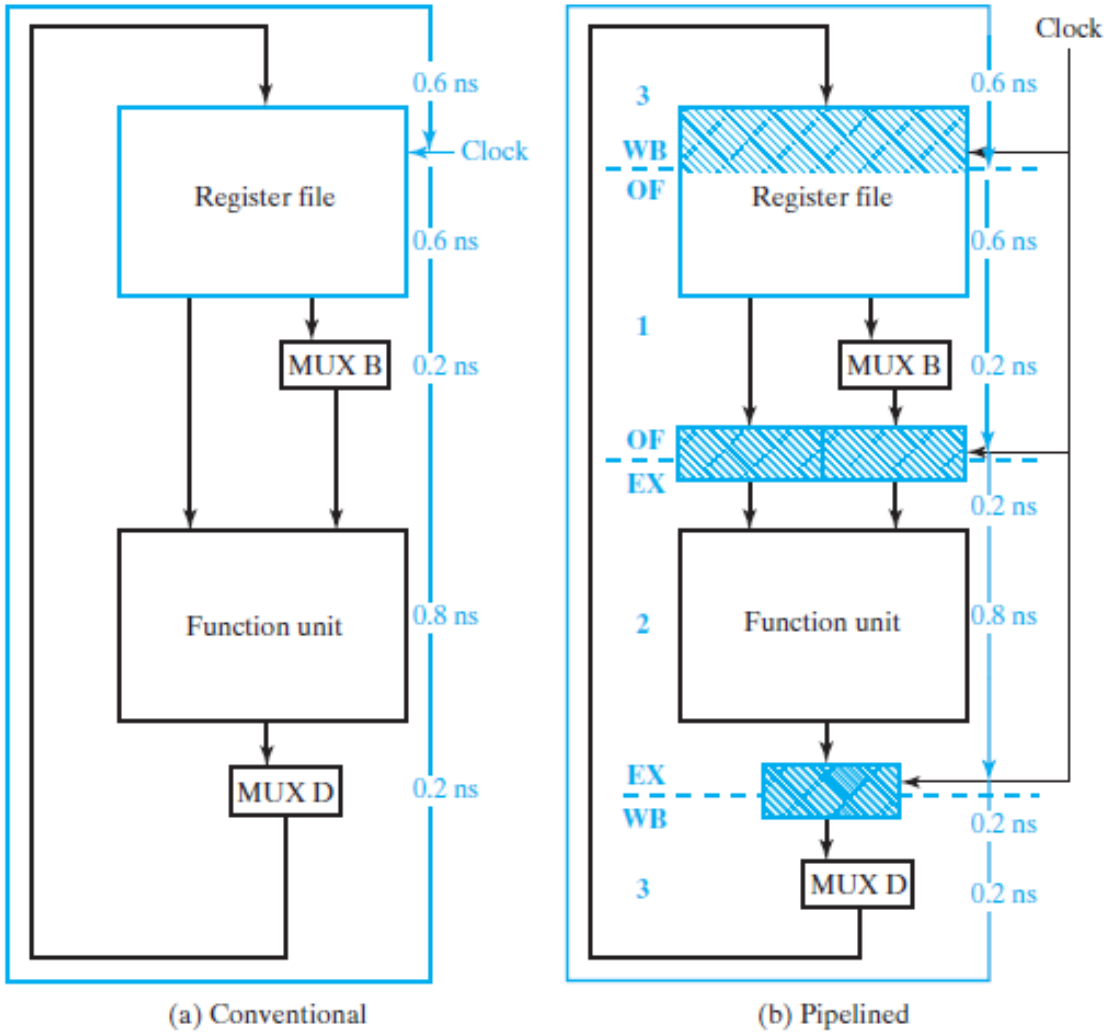


FIGURE 10-1
Datapath Timing

Pipelined Datapath

- This clock frequency corresponds to the maximum **throughput** of the pipeline, which is **1 billion instructions per second**, about 2.4 times that of the nonpipelined datapath. Even though there are three stages, the improvement factor is not 3.0 — for two reasons:

(1) the additional delay contributed by the pipeline platforms and

(2) the differences between the delay of the logic assigned to each stage.

- The clock period is governed by the longest delay, rather than the average delay of stages.
- Thus, when all three stages work in parallel, and different instructions are in different stages at the same time, the overall **speed improvement** can be **2.4 times** than non pipelined CPU.

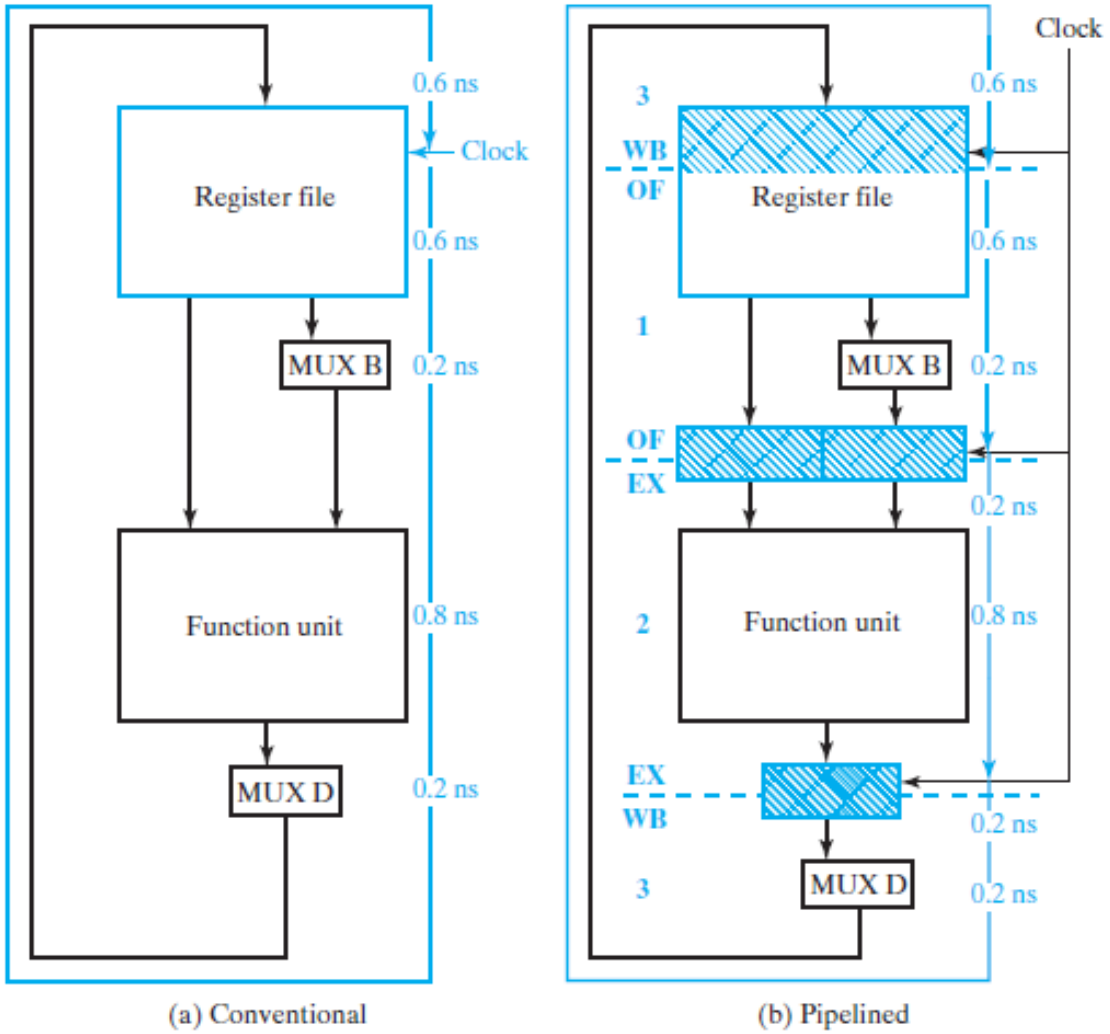


FIGURE 10-1
Datapath Timing

Pipelined Datapath

- A more detailed diagram of the pipelined datapath appears in Figure 10-2. In this diagram, rather than showing the path from the output of MUX D to the register file input, the register file is shown twice—once in the OF stage, where it is read, and once in the WB stage, where it is written.
- The first stage, **OF**, is the **operand fetch stage**. The operand fetch consists of reading register values to be used from the register file and, for Bus B, selecting between a register value or a constant by using MUX B.
- Following the OF stage is the first pipeline platform. The pipeline registers store the operand or operands for use in the next stage during the next clock cycle.
- The second stage of the pipeline is the **execution stage**, denoted **EX**. In this stage, a function unit operation occurs for most microoperations. The results produced from this stage are captured by the second pipeline platform.

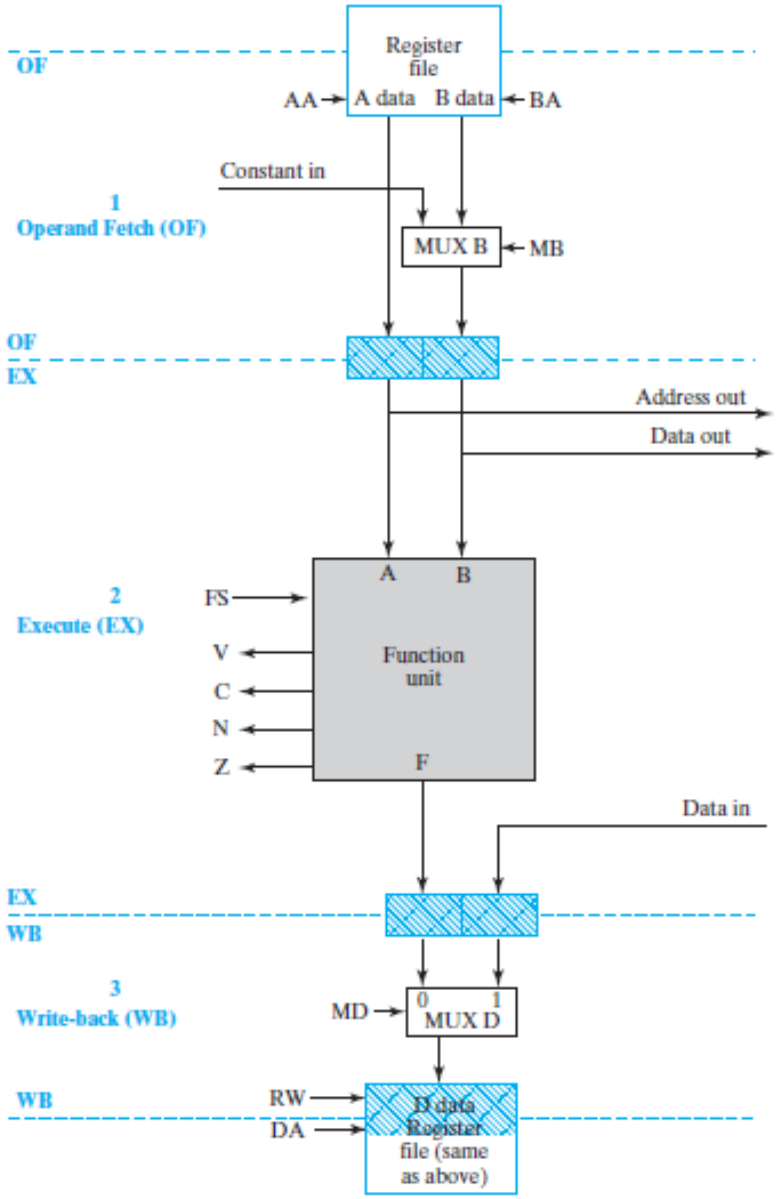


FIGURE 10-2
Block Diagram of Pipelined Datapath

Pipelined Datapath

- The third and final stage of the pipeline is the **write-back stage**, denoted **WB**. In this stage, the result saved from the EX stage, or the value on Data in, is selected by MUX D and written back into the register file at the end of the stage. In this case, the write part of the register file is the pipeline platform. The WB stage completes the execution of each microoperation that requires writing to a register.

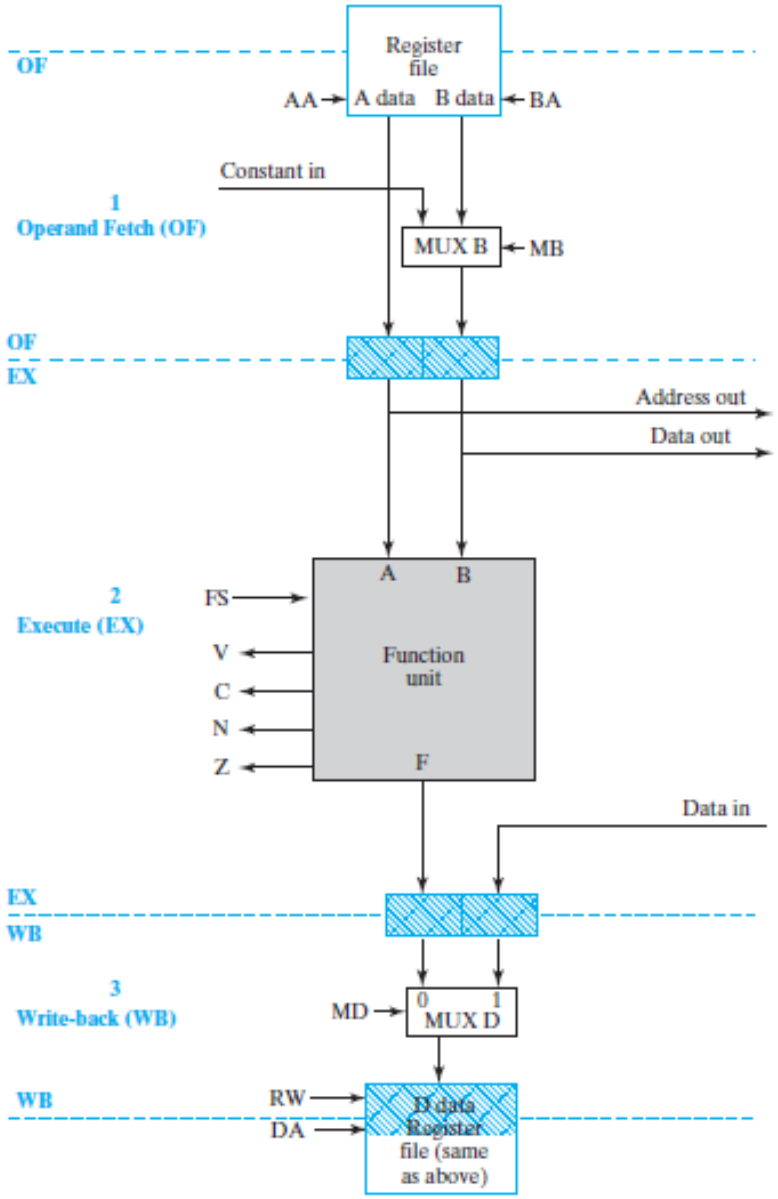


FIGURE 10-2
Block Diagram of Pipelined Datapath

Execution of Pipeline Microoperations

- We can have three microoperations at some stage of completion in the pipelined datapath at any given time.
- In clock period 1, microoperation 1 is in the OF stage. In clock period 2, microoperation 1 is in the EX stage, and microoperation 2 is in the OF stage. In clock period 3, microoperation 1 is in the WB stage, microoperation 2 is in the EX stage, and microoperation 3 is in the OF stage.
- So at the end of the third clock period, microoperation 1 has completed execution, microoperation 2 is two-thirds finished, and microoperation 3 is one-third finished. So we have completed 1 + 2/3 + 1/3 = 2.0 microoperations in three clock periods, or 3 ns.
- In the conventional datapath, we would have completed microoperation 1 only. So, indeed, the pipelined datapath performance is superior in this example.

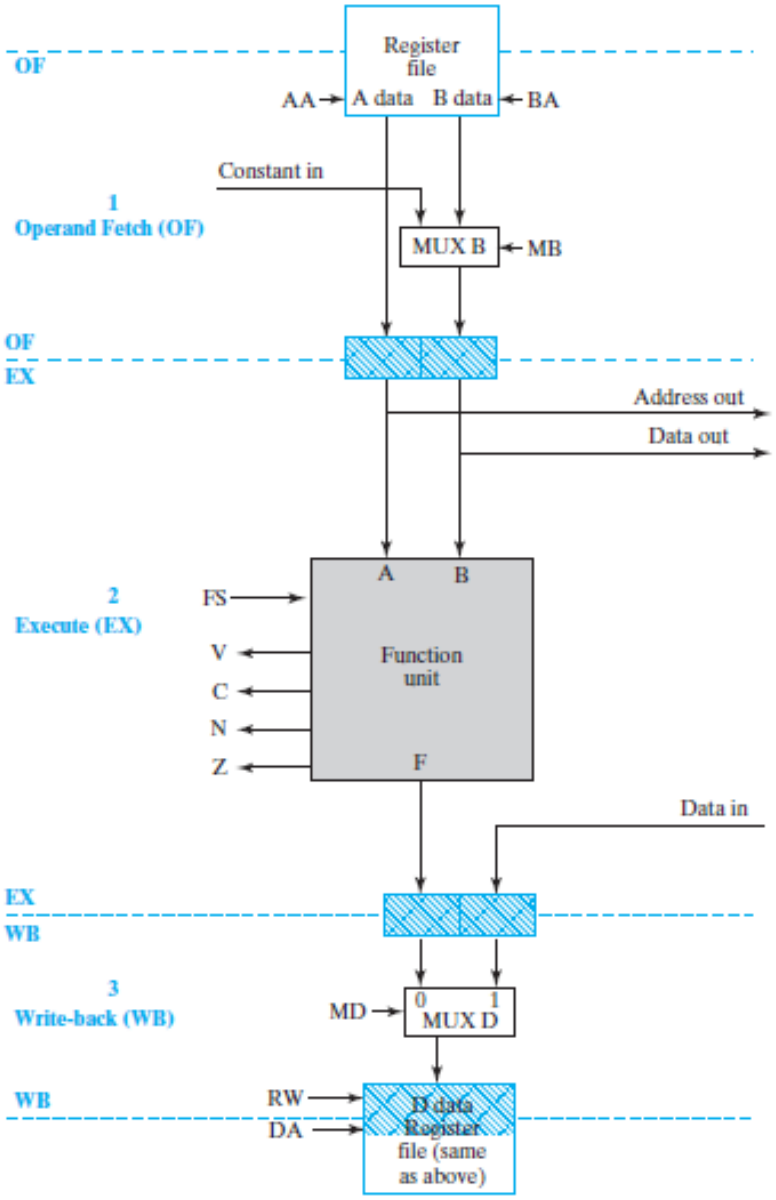


FIGURE 10-2
Block Diagram of Pipelined Datapath

Execution of Pipeline Microoperations

- To illustrate it better, we will use a pipeline execution pattern diagram, as shown in Figure 10-3.
- Each vertical position in this diagram represents a microoperation to be performed, and each horizontal position represents a clock cycle. An entry in the diagram represents the stage of processing of the microoperation. So, for example, the execution (EX) stage of microoperation 4, which adds the constant 2 to R0, occurs in clock cycle 5.

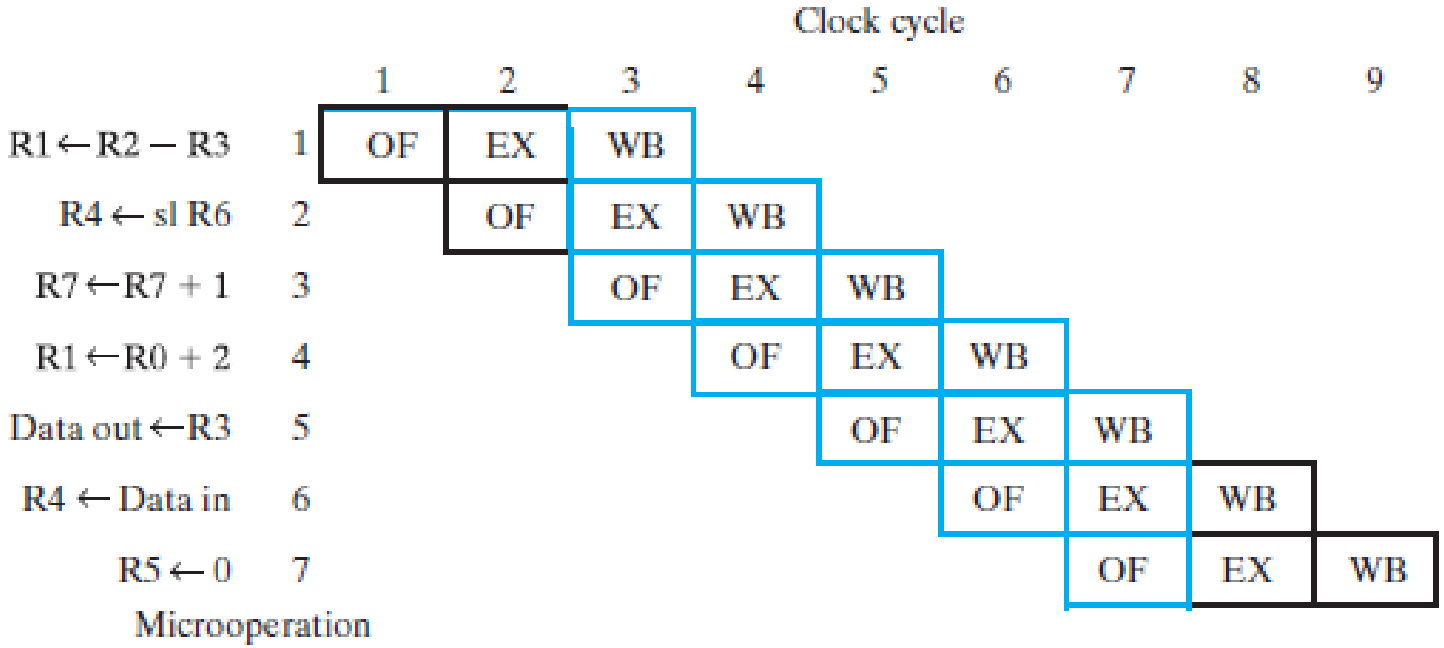


FIGURE 10-3
Pipeline Execution Pattern for Microoperation Sequence

Execution of Pipeline Microoperations

- We can see from the overall diagram that the sequence of seven microoperations requires nine clock cycles to execute completely. The time required for execution is 9 * 1 = 9 ns, compared to 7 * 2.4 = 16.8 ns for the conventional (not pipelined) datapath.
- Thus, the sequence of microoperations is executed about 1.9 times faster using the pipeline.
- Now let us examine the pipeline execution pattern carefully. In the first two clock cycles, not all of the pipeline stages are active, since the pipeline is filling. In the next five clock cycles, all stages of the pipeline are active, as indicated in blue, and the pipeline is fully utilized. In the last two clock cycles, not all stages of the pipeline are active, since the pipeline is emptying.
- If we want to find the maximum possible improvement of the pipelined datapath over the conventional one, we compare the two when the pipeline is fully utilized. Over these five clock cycles, 3 through 7, the pipeline executes (5 * 3) , 3 = 5 microoperations in 5 ns.
- In the same time, the conventional (not pipelined) datapath executes 5/2.4 = 2.083 microoperations. So the pipelined datapath executes at best 5 / 2.083 = 2.4 times as many microoperations in a given time as the conventional datapath. In this ideal situation, we say that the throughput of the pipelined datapath is 2.4 times that of the conventional one.

Pipelined CPU (from Single-Cycle)

- In this section, a control unit is discussed to produce a CPU by using the datapath from the last section.
- Since the instruction must be fetched from a memory as well as executed, we add a stage **instruction fetch or IF** to fetch the instruction from the instruction memory.
- Figure 10-4 shows the block diagram of a pipelined computer based on the single-cycle computer.
- The datapath is that of Figure 10-2. The control has an added stage for **instruction fetch (IF)** that includes the PC and instruction memory. This becomes stage 1 of the combined pipeline.
- The **instruction decoder and register file read (Decoder and Operand Fetch, DOF)** are now in stage 2, the **function unit and data memory read and write (EX)** are in stage 3, and the **register file write (WB)** is in stage 4.

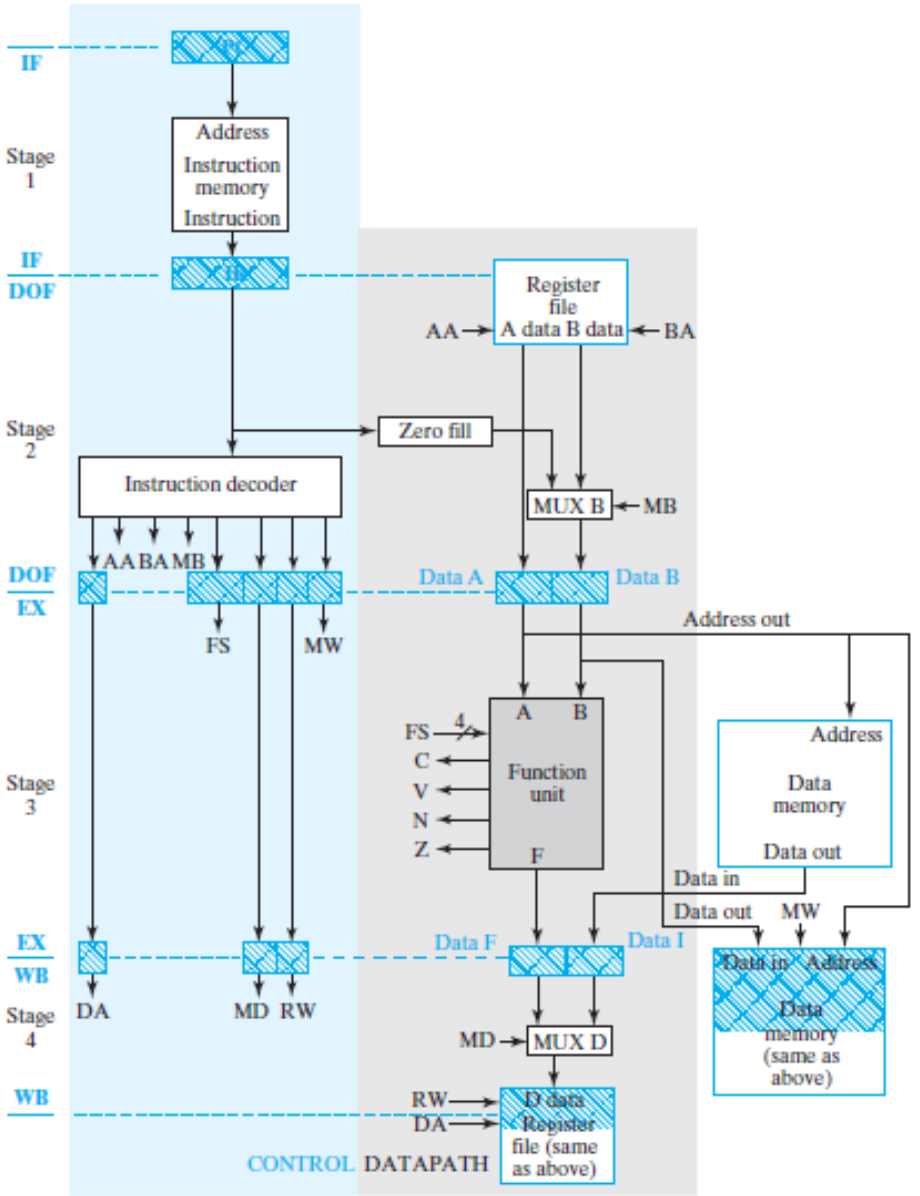


FIGURE 10-4
Block Diagram of Pipelined Computer

Pipelined CPU (from Single-Cycle)

- In the figure, registers are added to the pipeline platforms between stages, as necessary, to pass the decoded instruction information through the pipeline along with the data being processed.
- These additional registers serve to pass along the instruction information.
- The added first stage is the **instruction fetch stage**, denoted by **IF**. In this stage, the instruction is fetched from the instruction memory, and the value in the PC is updated.
- Due to additional complexities of handling jumps and branches in a pipelined design, PC update is restricted here to an increment, for now.
- Between the first stage and the second stage is an pipeline platform that plays the role of instruction register, so it has been labeled **IR**. IR receives Instruction from Instruction memory pointed by PC.

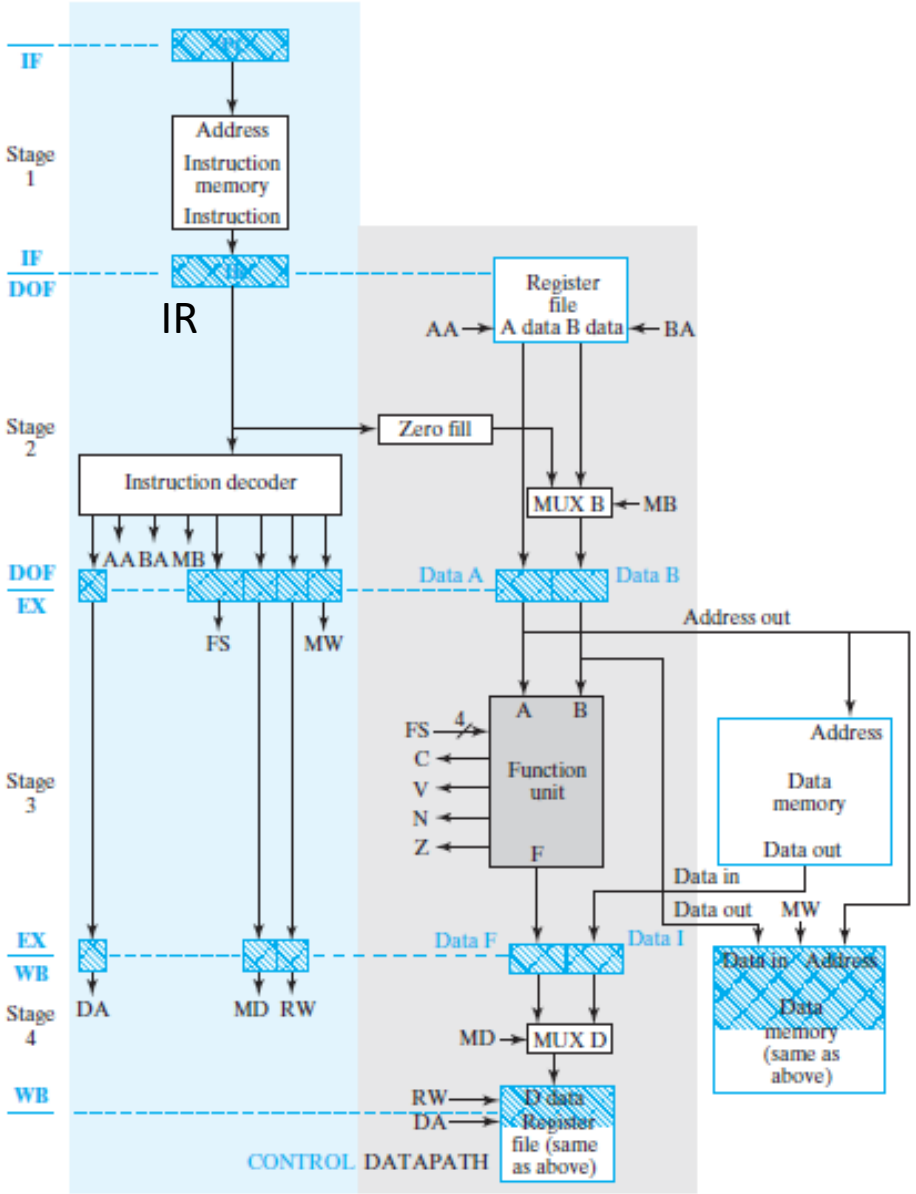


FIGURE 10-4
Block Diagram of Pipelined Computer

Pipeline CPU (from Single-Cycle)

- In the second stage, **DOF (decode and operand fetch)**, decoding of the IR into control signals takes place. Among the decoded signals, the register file addresses **AA** and **BA** and the multiplexer control signal **MB** are used in this stage for operand fetch. All other decoded control signals are passed on to the next platform, to be used later.
- The third stage of the pipeline is the **execution** stage, denoted **EX**. In this stage, an ALU operation, a shift operation, or a memory operation is executed for most instructions. Thus, the control signals used in this stage are **FS** and **MW**. The read part of the data memory M is a part of this stage. For a memory read, the value of the word addressed is read to Data out from the data memory. The write part of data memory can be a part of this platform, so a memory write may occur here.
- All of the results produced from one stage, plus the control signals for the last stage, are captured by the next pipeline platform.

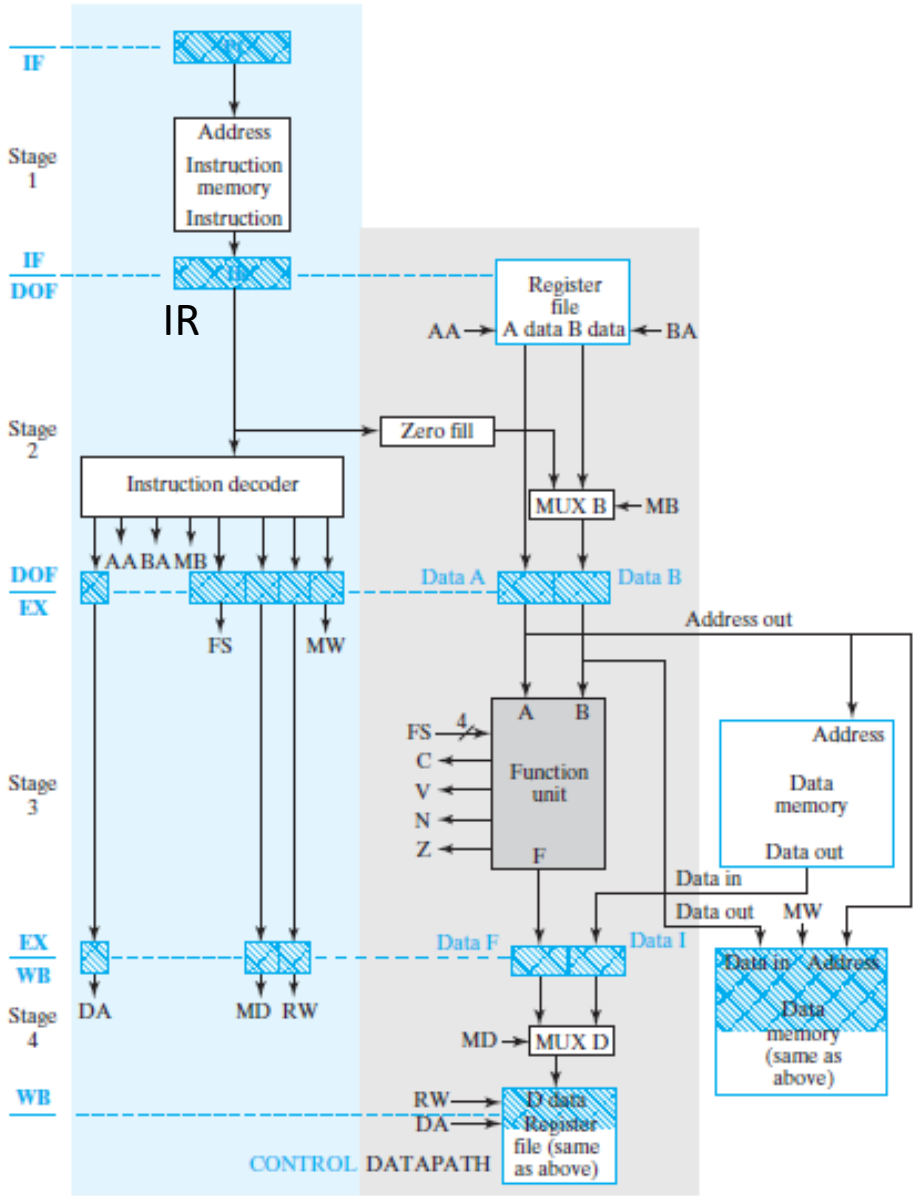


FIGURE 10-4
Block Diagram of Pipelined Computer

Pipeline CPU (from Single-Cycle)

- The control information held in the final pipeline platform consists of **DA**, **MD**, and **RW**, which are used in the final **write-back** stage, **WB**.
- The location of the pipeline platforms has balanced the partitioning of the delays, so that the delays per stage are no more than 1.0 ns. This gives a potential maximum clock frequency of 1 GHz, 3.4 times that of the single-cycle computer.
- Note, however, that an instruction takes 4 * 1 = 4 ns to execute. This latency of 4 ns compares to that of 3.4 ns for the single-cycle computer (without pipeline). So if only one instruction at a time is being executed, even fewer instructions are executed per second than for the single-cycle computer (pipelined).

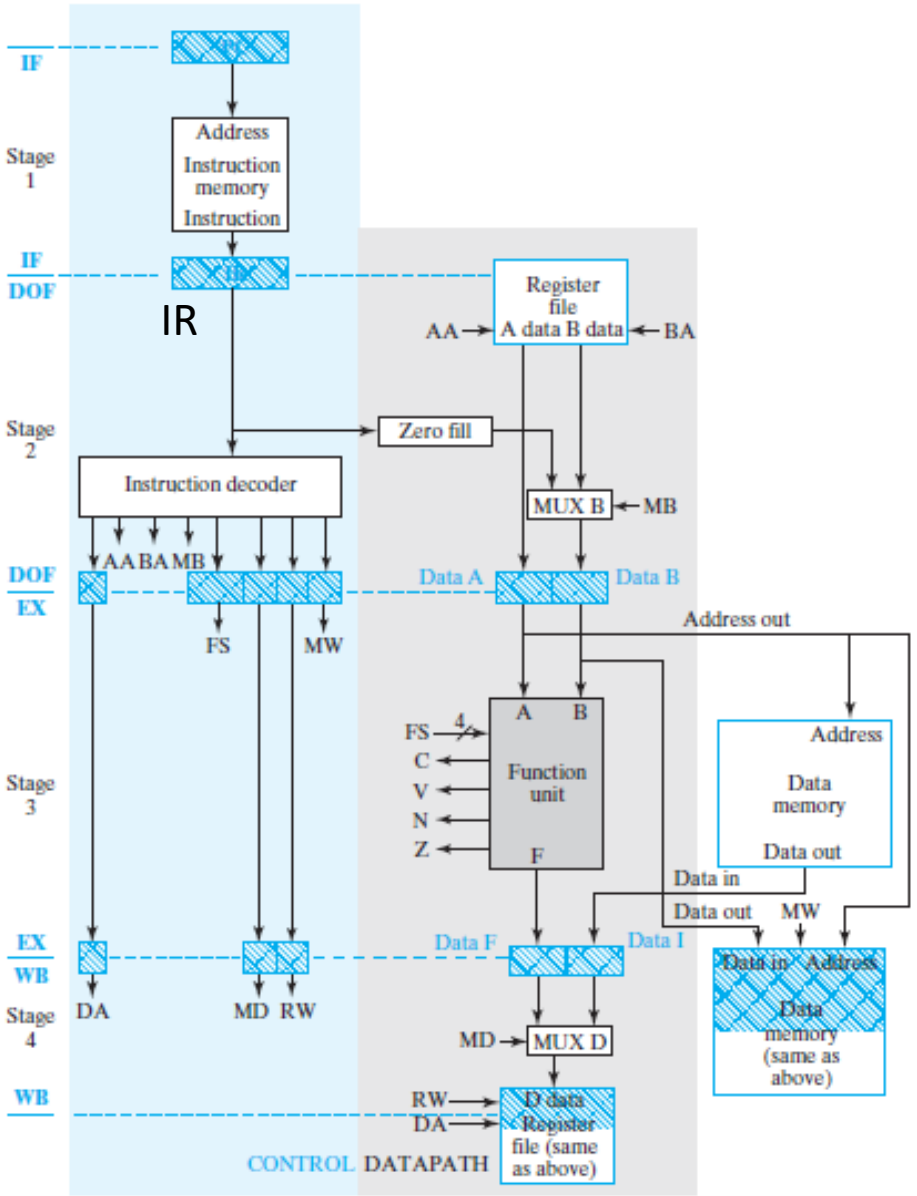


FIGURE 10-4
Block Diagram of Pipelined Computer

Pipeline Programming and Performance

- Suppose we consider a simple program: Load the constants 1 through 7 into the seven registers R1 through R7, respectively. The program to do this is as follows:

```
1        LDI R1, 1
2        LDI R2, 2
3        LDI R3, 3
4        LDI R4, 4
5        LDI R5, 5
6        LDI R6, 6
7        LDI R7, 7
```

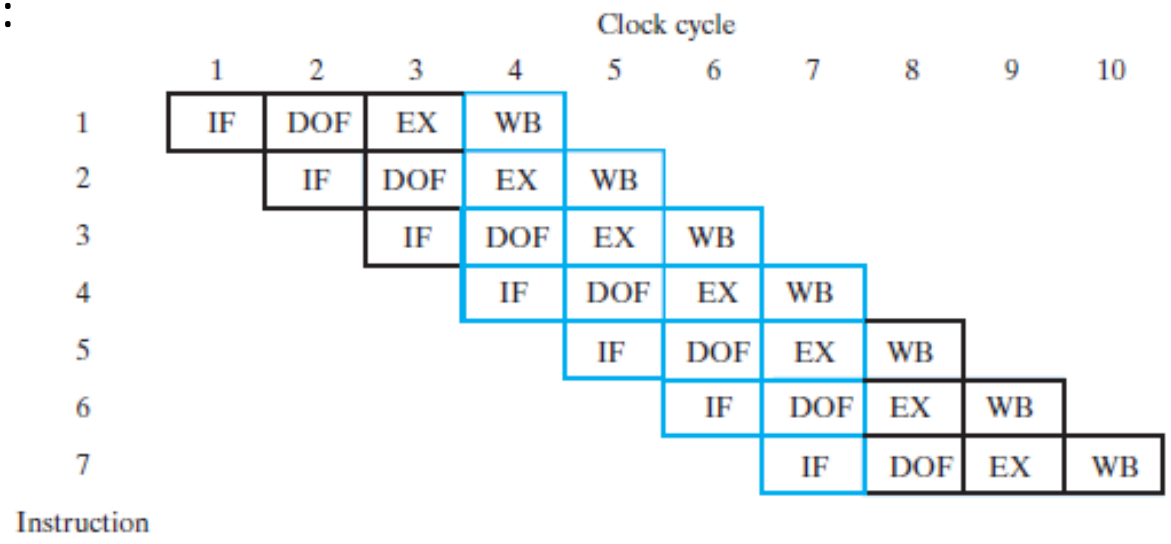


FIGURE 10-5
Pipeline Execution Pattern of Register Number Program

- Let us examine the execution of this program with respect to the stages of the pipeline(Figure 10-5).
- In clock period 1, instruction 1 is in the IF stage of the pipeline. In clock period 2, instruction 1 is in the DOF stage and instruction 2 is the IF stage. In clock period 3, instruction 1 is in the EX stage, instruction 2 is in the DOF stage, and instruction 3 is in the IF stage.

Pipeline Programming and Performance

- In clock period 4, instruction 1 is in the WB stage, instruction 2 is in the EX stage, instruction 3 is in the DOF stage, and instruction 4 is in the IF stage. So at the end of the fourth clock period, instruction 1 has completed execution, instruction 2 is three-fourths finished, instruction 3 is half finished, and instruction 4 is one-fourth finished.
- So we have completed 1 + 3/4 + 1/2 + 1/4 = 2.5.

2.5 instructions in four clock periods, or 4 ns. We can see from the overall diagram that the complete prograr of seven instructions requires 10 clock cycles to execute

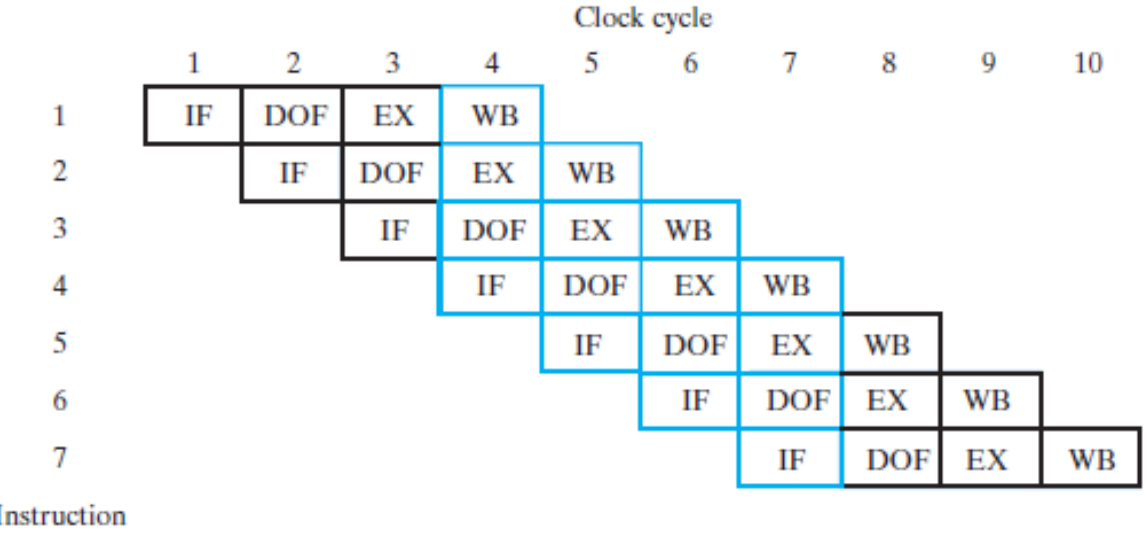


FIGURE 10-5
Pipeline Execution Pattern of Register Number Program

- Thus, the time required is 10 ns, compared to 23.8 ns for the single-cycle computer, and the program is executed about 2.4 times faster.
- Note that even though the pipeline has four stages, the pipelined computer is not four times as fast as the single-cycle computer, because of the additional delays and unequal times for each stage.

Assignment

- Go through slides 14, 15 and 16.
- Mention all the tasks done in every stage in Register Transfer Notation.
- The four stages are IF, DOF, EX and WB.
- For example, in IF stage, PC is incremented. So, it can be expressed as PC <- PC + 1
- DOF stage, to show that first operand appears on Bus A

  BusA <- R[AA]
- For multiple possibilities such as various values of FS, you need to write like that

  FS = 0000: F <- A

TABLE 8-4
***G* Select, *H* Select, and *MF* Select Codes Defined in Terms of *FS* Codes**

FS(3:0)	MF Select	G Select(3:0)	H Select(3:0)	Microoperation
0000	0	0000	XX	$F = A$
0001	0	0001	XX	$F = A + 1$
0010	0	0010	XX	$F = A + B$
0011	0	0011	XX	$F = A + B + 1$
0100	0	0100	XX	$F = A + \overline{B}$
0101	0	0101	XX	$F = A + \overline{B} + 1$
0110	0	0110	XX	$F = A - 1$
0111	0	0111	XX	$F = A$
1000	0	1X00	XX	$F = A \wedge B$
1001	0	1X01	XX	$F = A \vee B$
1010	0	1X10	XX	$F = A \oplus B$
1011	0	1X11	XX	$F = \overline{A}$
1100	1	XXXX	00	$F = B$
1101	1	XXXX	01	$F = \text{sr } B$
1110	1	XXXX	10	$F = \text{sl } B$