The Architecture of Computer Hardware and System Software
An Information Technology Approach
by Irv Englander

Chapter 7: The CPU and Memory

Outline

- The Components of the CPU
- The Concept of Registers
- The Memory Unit
- The Computer Operating Cycle (Fetch-Execute Instruction Cycle)
- Buses

Computer Hardware Components

Hardware: Input/Output, Memory, Central Processing Unit (CPU or Microprocessor)

System Block Diagram

7.1 The Components of the CPU

- CPU Consists of Several Integrated Circuits (ICs). For a Single Chip, CPU is Called: Microprocessor
- CPU Consists of
  - Set of Registers, SRs, which are used to store instructions, operands, and the result of the ALU
  - Arithmetic Logic Unit, ALU, which does all the arithmetic and most of the logical operations on operands
  - Control Unit, CU, which sequences the operation of the computer (Computer Cycle).
7.2 The Concept of Registers

- A Register is a single, permanent storage location within the CPU
- Registers may be small as a single bit or wide as several bytes
- Registers are used to hold data, instructions, I/O address, or special binary codes
- Registers inside the CPU are called: General-Purposes Registers
- There are some specific registers called:
  - Program Counter (PC)
  - Instruction Register (IR)
  - Memory Address Register (MAR)
  - Memory Data Register (MDR) (or MBR)
  - Status Register (ST)
  - I/O Registers

Operations on Registers

- Data can be loaded from register or memory into a register
- Data in register can be added/subtracted to/from another register
- Data in register can be shifted/rotated right/left by one or more bits
- Data in register can be cleared, set, complemented or incremented/decremented by one

Register is A Group of Flip-Flops

Output changes only on the rising edge of the clock
Simple Shift Register

<table>
<thead>
<tr>
<th>In</th>
<th>Q0</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>t0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>t3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>t5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>t6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>t7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Shift Register with Parallel Load

Parallel Load

Shift Register
Three-Bit Up-Counter

(a) Circuit

(b) Timing diagram

Four-Bit Synchronous Up-Counter

General CPU Bus Organization

Computer Organization

CPU Connection with Memory
7.3 The Memory Unit (RAM)

- A Memory Unit is a collection of storage cells together with associated circuits needed to transfer information in and out of storage.
- A Word is an entity of bits that move in and out of storage as a unit. It may represent a number, an instruction code, characters, or any other binary-coded information.
- A Byte is a group of eight bits.
- Memory capacity is the total number of bytes that can be stored.
- Number of words and the number of bits in each word represent the internal structure of the memory.

Random-Access Memory (RAM)

MAR: Memory Address Register holds the address of information to be stored/read from memory.

MDR: Memory Buffer Register holds the data/instruction to be stored/read from memory.

If the MAR size is \( k \), then the total number of words which can be addressed is \( m = 2^k \)

- 1K of RAM = \( 1024 = 2^{10} \)
- 1M of RAM = \( 1024 \times 1024 = 2^{20} \)
- 1G of RAM = \( 1K \times 1M = 2^{30} \)

MDR, MAR, and memory

MAR-MDR example
7.4 The Fetch-Execute Instruction Cycle

- Program to be executed by computer consists of sequence of instructions stored in memory
- The CPU executes each instruction of the program according to a predefined steps
- The Control Unit of the CPU determines these defined steps according to the Instruction Format and the Computer Architecture
- A program instruction must contain field for operation code, field for the address of the operands, and field to show where the operands are stored, address modification (they will be discussed in Chapter 8)

Instruction Codes

- An instruction code is a group of bits that instruct the computer to perform a specific operation.
- It is divided into three parts:
  - 1. The Operation Code: ADD, SUB, MULT, DIV, etc. The number of bits determines the number of operations. Example: 4-bit size produces 16 Opcodes
  - 2. The Address Code: the address of the two operands
  - 3. The Address Modification Code: where are the operands stored

Instruction Format:

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Operands Address</th>
<th>Address Mod</th>
</tr>
</thead>
</table>

General Form of A Computer Cycle

- Fetch: Read an Instruction From Memory
- Decode: Decode the Instruction; Obtain the Operation Code and the Operand Address
- Effective Address (EA): Calculate the Address of the Operands
- Execute: Perform the Required Operation and Store the Result in memory/register
- Repeat Until the end of Instructions
The Instruction must be Fetched from Memory
Read and Write Cycle

- **Write to Memory:**
  - T0 : AR <- Address, BR <- Data
  - T1 W: M[AR] <- BR

- **Read from Memory:**
  - T0 : AR <- Address
  - T1 R: BR <- M[AR]

- The Read/Write Cycle is written in a language called:
  Register Transfer Language
  T0 & T1 are Timing Signals
  AR: Address Register, BR: Buffer Register

Register Transfer Language

- **A Digital System** is an interconnection of Digital hardware Modules that accomplish a specific information-processing task
- **Digital Modules** are best defined by the registers they contain and the operations that are performed on the data stored in them
- **Micro-operation** is an elementary operation performed on the information stored in one or more registers. **Examples** are shift, clear, and load operations

Register Transfer Language continue

- The internal Hardware Organization of a digital computer is best defined by specifying:
  1. the set of registers it contains and their function,
  2. the sequence of microoperations performed, and
  3. the control that initiates the sequence of microoperations
- **A Register Transfer Language** is a system for expressing in Symbolic Form the microoperation sequences among the registers of a digital system
- It is a tool for describing the internal organization of a digital computer in concise and precise manner

Tri-state Buffer

<table>
<thead>
<tr>
<th>C</th>
<th>A</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
In-Out control

Register Transfer

Register Transfer (continue)

Instruction Execution

- The Program Counter, PC, contains the address of the first instruction of the program.
- The first step in the execution of every instruction will be (put the address of the instruction into the memory address register)
  \[ T_0 : \text{MAR} \leftrightarrow \text{PC} \]
- The second step will be (read the instruction from memory and put it in memory data register)
  \[ T_1 : \text{MDR} \leftarrow \text{M[MAR]}, \text{PC} \leftarrow \text{PC} + 1 \]
- The third step will be (move the instruction from MDR to the instruction register)
  \[ T_2 : \text{IR} \leftarrow \text{MDR} \]
**Instruction Execution**

- The fourth step will be (decode the instruction and determine the address of the operands)
  
  \[ T_3 : \text{Op-Code} \leftarrow IR[\text{Op-Code}], \text{Address} \leftarrow IR[\text{Address}] \]

- The fifth step will be (execute the instruction which is instruction dependent on the op-code such as Load, Add,... etc). In this case, the operands (data) must be read from memory or they should be stored in registers as shown
  
  \[ T_4 : A \leftarrow M[\text{Address}] \]

- Load the operand from memory and put it in the accumulator register A

---

**Levels of Representation (61C Review)**

| Program Level | Example
|---------------|--------|
| High Level Language | temp = v[k];
| Assembly Language | v[k] = v[k+1];
| Machine Language | v[k+1] = temp; |

- lw $15, 0($2)
- lw $16, 4($2)
- sw $16, 0($2)
- sw $15, 4($2)


---

**What’s a Clock Cycle?**

- Old days: 10 levels of gates
- Today: determined by numerous time-of-flight issues + gate delays
  - clock propagation, wire lengths, drivers

---

- Instruction 00
Beginning, Program & Data in Memory
Reset counter, the Machine in a random state

Instruction Fetch (1) ...
PC → MAR, Read

Instruction Fetch (2) ...
MDR → IR, PC↑1

Instruction Execute ...
In → Accu
Next instruction: 01
Next instruction: 02
Next instruction: 03
Next instruction: 04
Instruction Execute (2) ...
ACCU - MDR → ACCU

Next instruction: 05

Instruction Fetch (1) ...
PC → MAR, Read

Instruction Fetch (2) ...
MDR → IR, PC↑1
Instruction Execute ...
(acc ≥ 0 ⇒ IR[adress] → PC)

Next instruction: 08
Instruction Execute ...
ACCU → OUT

Instruction Fetch (1) ...
PC → MAR, Read

Instruction Fetch (2) ...
MDR → IR, PC↑1

Next instruction: 09
### 7.5 Buses

- A Bus is a group of electrical conductors suitable for carrying computer signals from one location to another.
- Buses are used to transfer data/instructions among computer peripherals, memory and the CPU.
- The lines on a bus can be grouped into four categories:
  - Data,
  - Addressing,
  - Control, and
  - Power.
Different Buses for Connecting Different parts of the Computer

• The buses internal to the CPU don’t have a names
• The CPU bus, Peripheral Connect Interface (PCI) or Video Electronic Standards Association (VESA) local bus, and Industry Standard Architecture (ISA) bus are all part of the Backplane as shown
• The PCI, VESA local, also known as VL, and ISA buses are examples of popular modern external buses
• A bus protocol is an agreement between two or more entities that establish a clear, common path of communication and understanding between them
• Buses can be characterized by:
  – Throughput (Data Transfer Rate, bit/second),
  – the Data Width in bits,
  – the distance between the two end points,
  – the type of control required,
  – the type of bus,
  – the addressing capacity, and
  – the dedicated line or shared line inside the bus

7.6 Timing Issues

• The time of the events at the CPU is synchronized to the pulses of an electronic clock
• The clock provides a master control as to when each step in the instruction cycle takes place
• The pulses in the clock is separated sufficiently to assure that each step has enough time to complete, with the data settled down, before the results of that step are required by the next step
• Wait state is provided when the clock pulse is not wide enough (slow down the speed of the computer)
• Each clock pulse is used to control one step in the sequence of the instruction execution
• Sometime, additional pulses might be used to control different details within a single step