Processor and Computer

John SUM
Institute of Technology Management
National Chung Hsing University
Taichung, ROC

May 15, 2020

Contents

Conceptually, a processor (or CPU) is a complex digital logic circuit which consists of many small digital logic circuits, registers and switches (resp. connectors). Each small digital logic circuit performs a simple logical operation. The registers are used as temporary working memory space to support the completion of an instruction. The switches (resp. connectors) are used for controlling the flow of signal amongst the registers, the logic gates and the logic circuits.

1 Instruction Execution

Basically, a processor will go through two phases once an instruction has been fed in. The first phase is called *instruction decode*. In this phase, the *control unit* will decode the instructions to a sequence of binary signals, called the *micro-instruction*, for controlling the switches (resp. connectors) which are used for controlling the flow of signal amongst the registers, the logic gates and the logic circuits. The second phase is called *execution*. The control unit generates the micro-instructions, one micro-instruction per clock cycle, for controlling the switches and connectors.

2 Simple Processor

Figure 1 shows a simple processor with four logic gates, seven registers and eighteen switches. Registers RA, RB and RZ are registers associated with the logic gates, while the registers $R1$, $R2$, $R3$ and $R4$ act as temporary memory spaces.

2.1 Switches

The connections between the set of registers (RA, RB, RZ) and the set of logic gates are controlled by simple switches, called connectors, S_1, S_2, $\cdots$, S_{11}. Each connector connects two metal lines if the signal received is '1'. Otherwise, the connector disconnects the lines. By default, a connector is in the condition of disconnection.

$$\text{Connection} = \begin{cases} \text{Connect} & \text{if } S_i = 1, \\ \text{Disconnect} & \text{if } S_i = 0. \end{cases}$$

for $i = 1, \cdots, 11$.

The flow of data amongst the registers $\{RA, RB, RZ, R1, R2, R3, R4\}$ and the common connection line is controlled by two-way switches $S_{12}, S_{13}, \cdots, S_{18}$. Each two-way switch is associated with a single register to control if a data is to be read from or written to a register.

S_i	Action
00	Disconnect.
01	Data flowing out of the register, i.e. read data from the register.
10	Data flowing in the register, i.e. write data to the register.

2.2 Micro-instruction

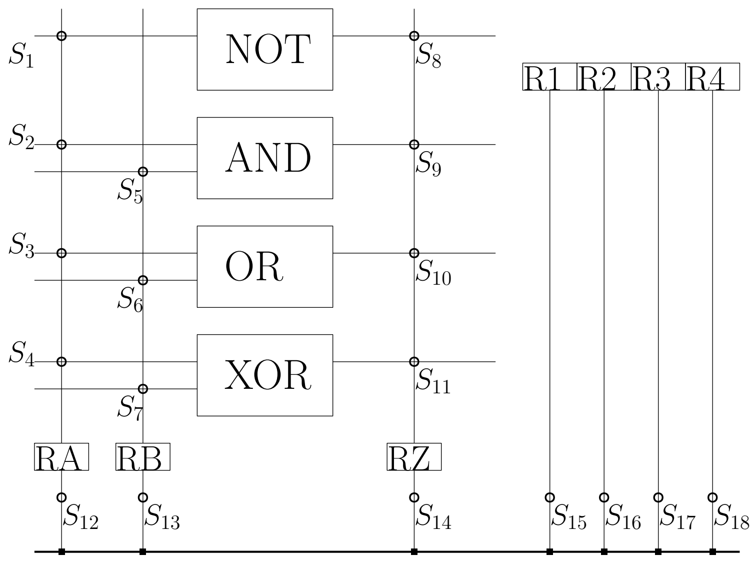
To complete a task specified by an instruction, the control unit needs to decode the instruction into a sequence of micro-instructions. Figure 2 shows a sample of eight instructions and their micro-instructions. The first five instructions are the simplest instructions. Each of these simple instructions consists of only one micro-instruction. For the other three instructions are more complicated instructions.

Here, one should note that the design of the micro-instructions for an instruction is depended on the architecture of the processor. The micro-instructions depicted in Figure 2 are designed based on the processor architecture as shown in Figure 1. It consists of four additional registers, $R1$, $R2$, $R3$ and $R4$, as the working memory space for completion of the task specified by an instruction. In this design, it is allowed to move a data within the registers $R1$, $R2$, $R3$ and $R4$.

2.3 Register access

Based upon the architecture as shown in Figure 1, the total number of electrical signals to be generated to control the read/write of the registers $R1$, $R2$, $R3$ and $R4$ are eight. If the number of registers is scaled up to n, the number of electrical signals to be generated will be $2n$. For large n, this number will be large.

Figure 3 shows an alternative and yet simplified architecture for the access of registers $R1$, $R2$, $R3$ and $R4$. Instead of using four two-way switches for the registers, the registers are connected to the data line via three different switches. The empty (resp. solid) black circle switch is 'ON' if the control signal is '1' (resp. '0'). These switches are connected to two signal generators



Two-Way Switches: Disconnected (00); Down (01), Up (10).

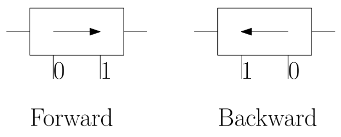
$(S_{12}, S_{13}, S_{14}, S_{15}, S_{16}, S_{17}, S_{18})$

Connectors: Disconnected (0), Connected (1).

$(S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11})$

(a) Processor

Two-Way Electronic Switch



(b) Two-Way switch (resp. R/W switch)

Figure 1: A processor with four logic gates. Switches S_1 to S_{11} are simple switches (i.e. connectors). S_{12} to S_{18} are two-way switches. '01' also refers to the 'read' action. Data is read from the register. '10' refers to the 'write' action. Data is pass to the register.

Instructions	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}	S_{17}	S_{18}
MOV RA R1	0	0	0	0	0	0	0	0	0	0	0	10	00	00	01	00	00	00
NOT RA	1	0	0	0	0	0	0	1	0	0	0	00	00	00	00	00	00	00
AND RA RB	0	1	0	0	1	0	0	0	1	0	0	00	00	00	00	00	00	00
OR RA RB	0	0	1	0	0	1	0	0	0	1	0	00	00	00	00	00	00	00
XOR RA RB	0	0	0	1	0	0	1	0	0	0	1	00	00	00	00	00	00	00
NAND RA RB	0	1	0	0	1	0	0	0	1	0	0	00	00	00	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	10	00	01	00	00	00	00
	1	0	0	0	0	0	0	1	0	0	0	00	00	00	00	00	00	00
$A + AB$	0	1	0	0	1	0	0	0	1	0	0	00	00	00	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	00	10	01	00	00	00	00
	0	0	1	0	0	1	0	0	0	1	0	00	00	00	00	00	00	00
$(\neg A + B)(A + \neg B))$	0	0	0	0	0	0	0	0	0	0	0	01	00	00	10	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	00	01	00	00	10	00	00
	1	0	0	0	0	0	0	1	0	0	0	00	00	00	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	00	00	01	00	00	10	00
	0	0	0	0	0	0	0	0	0	0	0	10	00	00	00	01	00	00
	1	0	0	0	0	0	0	1	0	0	0	00	00	00	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	01	00	00	00	00	00	10
	0	0	0	0	0	0	0	0	0	0	0	00	10	00	00	00	01	00
	0	0	1	0	0	1	0	0	0	1	0	00	00	00	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	00	00	01	00	00	10	00
	0	0	0	0	0	0	0	0	0	0	0	10	00	00	01	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	00	10	00	00	00	00	01
	0	0	1	0	0	1	0	0	0	1	0	00	00	00	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	00	10	01	00	00	00	00
	0	0	0	0	0	0	0	0	0	0	0	10	00	00	00	00	01	00
	0	1	0	0	1	0	0	0	1	0	0	00	00	00	00	00	00	00

Figure 2: Sample micro-instructions for the processor with architecture as shown in Figure 1.

$A1$ and $A2$. The switches connecting the R/W are two-way switches. Thus, the data to be read from and written to the registers $R1$, $R2$, $R3$ and $R4$ could be defined as in the following table.

$A1$	$A2$	R/W	Action
0	0	01	Read data from $R1$
0	1	01	Read data from $R2$
1	0	01	Read data from $R3$
1	1	01	Read data from $R4$
0	0	10	Write data to $R1$
0	1	10	Write data to $R2$
1	0	10	Write data to $R3$
1	1	10	Write data to $R4$
x	x	00	Disconnection

In this regard, the number of signal to be generated reduces from $2n$ to $\log_2(n) + 2$. By using this new design for the access of the registers $R1$, $R2$, $R3$ and $R4$, the micro-instructions depicted in Figure 2 could be re-designed, as shown in Figure 4.

2.4 Instruction set

From the above two different architectures, one can expect that their instruction sets have a subtle difference. For the processor with architecture shown in Figure 1, it is allowed to move a data within the registers $R1$, $R2$, $R3$ and $R4$ in clock cycle. For the processor with architecture shown in Figure 5, it is not possible to move a data within the registers $R1$, $R2$, $R3$ and $R4$ in clock cycle.

If we consider an instruction as the instruction which can be completed in one clock cycle, the set of MOV instructions provided by the processor as shown in Figure 1 will be different from the set of MOV instructions provided by the processor as shown in Figure 5. Their differences are depicted in Table 1.

3 Memory Access

As a matter of fact, the design for register access could also be applied to design memory access, Figure 6. $MA1$ and $MA2$ refer to the memory address.

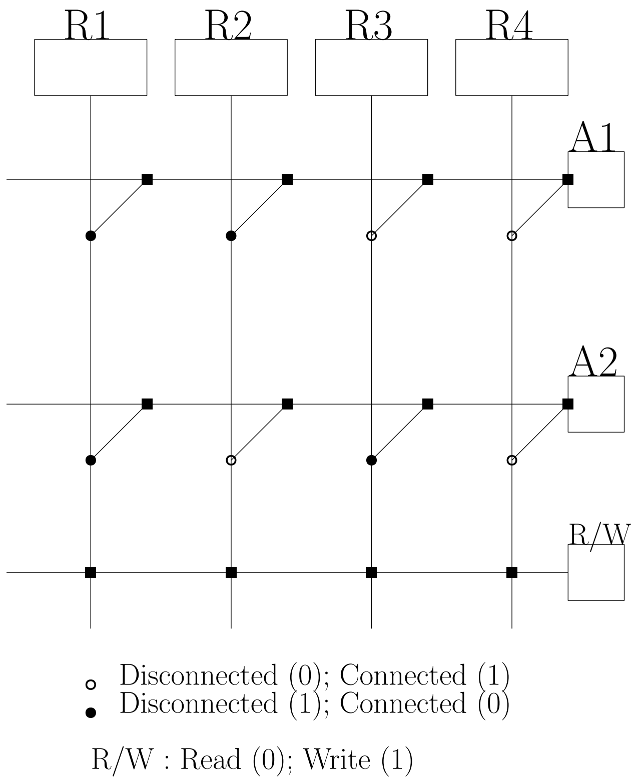

R1
R2
R3
R4
A1
A2
R/W
○ Disconnected (0); Connected (1)
● Disconnected (1); Connected (0)
R/W : Read (0); Write (1)

Figure 3: To read data from and write to a register, electrical signal corresponding to the register address is needed. Moreover, electrical signal indicating the read or write action is also needed to be sent to the two-way switches. This signal is called the R/W signal.

Instructions	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	A_1	A_2	R/W
MOV RA R1	0	0	0	0	0	0	0	0	0	0	0	10	00	00	0	0	01
NOT RA	1	0	0	0	0	0	0	1	0	0	0	00	00	00	0	0	00
AND RA RB	0	1	0	0	1	0	0	0	1	0	0	00	00	00	0	0	00
OR RA RB	0	0	1	0	0	1	0	0	0	1	0	00	00	00	0	0	00
XOR RA RB	0	0	0	1	0	0	1	0	0	0	1	00	00	00	0	0	00
NAND RA RB	0	1	0	0	1	0	0	0	1	0	0	00	00	00	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	10	00	01	0	0	00
	1	0	0	0	0	0	0	1	0	0	0	00	00	00	0	0	00
$A + AB$	0	1	0	0	1	0	0	0	1	0	0	00	00	00	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	00	10	01	0	0	00
	0	0	1	0	0	1	0	0	0	1	0	00	00	00	0	0	00
$(\neg A + B)(A + \neg B))$	0	0	0	0	0	0	0	0	0	0	0	01	00	00	0	0	10
	0	0	0	0	0	0	0	0	0	0	0	00	01	00	0	1	10
	1	0	0	0	0	0	0	1	0	0	0	00	00	00	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	00	00	01	1	0	10
	0	0	0	0	0	0	0	0	0	0	0	10	00	00	0	1	01
	1	0	0	0	0	0	0	1	0	0	0	00	00	00	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	01	00	00	1	1	10
	0	0	0	0	0	0	0	0	0	0	0	00	10	00	1	0	01
	0	0	1	0	0	1	0	0	0	1	0	00	00	00	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	00	00	01	1	0	10
	0	0	0	0	0	0	0	0	0	0	0	10	00	00	0	0	01
	0	0	0	0	0	0	0	0	0	0	0	00	10	00	1	1	01
	0	0	1	0	0	1	0	0	0	1	0	00	00	00	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	00	10	01	0	0	00
	0	0	0	0	0	0	0	0	0	0	0	10	00	00	1	0	01
	0	1	0	0	1	0	0	0	1	0	0	00	00	00	0	0	00

Figure 4: Sample micro-instructions for the processor with architecture as shown in Figure 1 with the new register access design as shown in Figure 3 and Figure 5.

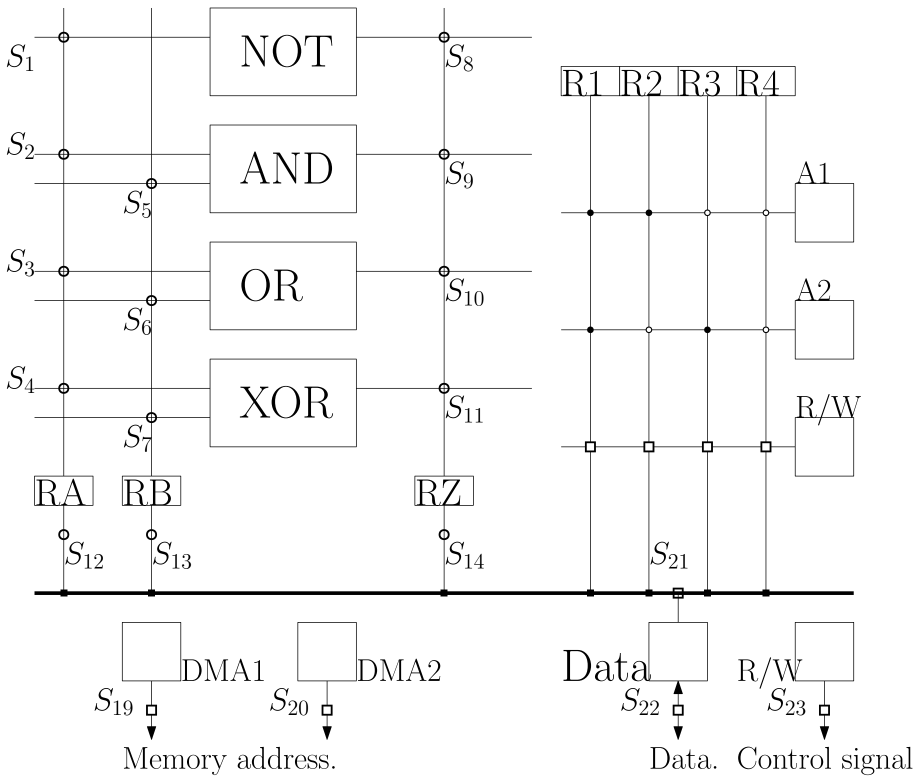


Figure 5: A processor with four logic gates and a new design for register access. Switches S_1 to S_{11}, S_{19}, S_{20} and S_{23} are simple switches (i.e. connectors). S_{21}, S_{22} and the switches connected to the R/W signal for register access are two-way switches.

Table 1: Different instruction sets for different architectures.

Processor in Figure 1	Processor in Figure 5
MOV RA RZ	MOV RA RZ
MOV RB RZ	MOV RB RZ
MOV RN RA	MOV RN RA
MOV RN RB	MOV RN RB
MOV RN RZ	MOV RN RZ
MOV RA RN	MOV RA RN
MOV RB RN	MOV RB RN
MOV RN RM	–

RN, RM refers to $R1$, $R2$, $R3$ and $R4$.

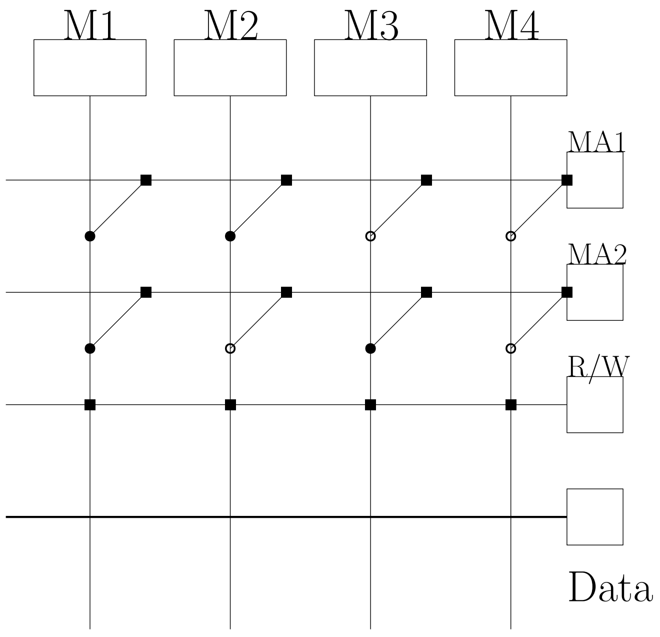
Once the binary signals have been fed to these two points, the corresponding memory location will have to take action. Data will be read from (resp. written to) the memory location if the R/W signal is 01 (resp. 10).

$MA1$	$MA2$	R/W	Action
0	0	01	Read data from $M1$
0	1	01	Read data from $M2$
1	0	01	Read data from $M3$
1	1	01	Read data from $M4$
0	0	10	Write data to $M1$
0	1	10	Write data to $M2$
1	0	10	Write data to $M3$
1	1	10	Write data to $M4$

3.1 Memory address

For a memory with n bits memory address, i.e. $MA1, \cdots, MAn$, the maximum number of memory spaces to be accessed is 2^n. By convention, the address of the first memory location is numbered 'zero'[1].

[1] In this lecture note, the first memory location is always named $M1$. As a matter of fact, the names for the memory locations should be started with $M0$. That is to say, the names for the memory locations should be $M0$, $M1$, $M2$ and so on.



○ Disconnected (0); Connected (1)
● Disconnected (1); Connected (0)

R/W : Read (0); Write (1)

Figure 6: To read a data from a memory location, (i) electrical signal corresponding to the memory address and (ii) a signal '0' will be sent from the processor to $MA1$ and $MA2$; and the R/W line of the memory. Thus, the data located in the corresponding memory location will be available at the data line. As the data line is connected to the data line of the processor. The data could thus be latched in the next clock cycle to the processor. To write a data to the memory, (i) electrical signal corresponding to the memory address, (ii) a signal '1' and (iii) the data will be sent from the processor to $MA1$ and $MA2$; the R/W line and the data line of the memory.

$MA2$	$MA1$	Memory location
0	0	$M1$
0	1	$M2$
1	0	$M3$
1	1	$M4$

In a processor, the memory address is generated by the control unit and stored in a set of special registers. Let us call them $DMA1$, $DMA2$ and so on. Again, each of these registers is connected to a two-way switch to a metal pin connecting to the memory device. The metal pin corresponding to the $DMAi$ ($i = 1, \cdots, n$) will be connected to the MAi pin of the memory device.

3.2 Data register

In principle, a processor has another special register called 'Data'. Whenever the content of a register has to be written to the main memory, its content will be temporary stored in the Data register. Similar to that of the registers for memory address, the Data register is connected to a two-way switch to a metal pin connecting to the memory device.

3.3 Move data from/to memory

With the above concepts, we could describe the mechanism how to write the content of a register, say RZ, to the memory location, say $M1$. In assembly language, the instruction is *MOV M1 RZ.*

1. Control unit signals $S_{14} = 01$ and $S_{21} = 10$. Other switches are set to disconnection mode.

2. Control unit signals $S_{19} = S_{20} = 1$, $S_{22} = 01$, $DMA1 = 0$, $DMA2 = 0$, $R/W = 10$ (the one which is next to the Data register) and $S_{23} = 1$. Other switches are set to disconnection mode.

3. Since $MA1 = DMA1 = 0$, $MA2 = DMA2 = 0$ and $R/W = 10$, the data in 'Data' is copied to the memory location $M1$.

Similarly, we could describe the mechanism how to read the content of $M1$ to RA. Equivalently, the instruction is *MOV RA M1.*

1. Control unit signals $S_{19} = S_{20} = 1$, $S_{22} = 10$, $DMA1 = 0$, $DMA2 = 0$, $R/W = 01$ (the one which is next to the Data register) and $S_{23} = 1$. Other switches are set to disconnection mode.

2. Since $MA1 = DMA1 = 0$, $MA2 = DMA2 = 0$ and $R/W = 01$, the content of $M1$ is moved to 'Data' in the memory device. As $S_{22} = 10$, the 'Data' is transferred to the 'Data' in the processor simultaneously.

3. Control unit signals $S_{12} = 10$ and $S_{21} = 01$. Other switches are set to disconnection mode.

3.4 Access time

Generally speaking, memory access time refers the time to read/write a data from/to the memory. Usually, different memory devices would have different access times.

- The access time between two registers in a processor is the smallest. It is in an order of magnitude 10^{-9} second.
- The access time between a processor register and a dynamic RAM (DRAM) is in a similar order of magnitude 10×10^{-9} second.
- The access time between a processor register and a static RAM (SRAM) is around 50×10^{-9} to 150×10^{-9} second.
- The access time between a processor register and a solid state drive (SSD) is in the range of 25×10^{-6} to 100×10^{-6} second.
- The access time between a processor register and a typical hard disk is in the range of 5×10^{-3} to 10×10^{-3} second.

In view of the access time, one should realize the reasons why RAM is needed. Once a processor is executing an instruction, it is likely that the number of registers inside the processor is not enough. Recall that some registers in a processor is grouped together as a cache. The cache acts as a temporary working space for the processor.

If this working space is not large enough, the only solution is to swap out some data in the cache to the hard disk and later swap them back to the processor for completion of the instruction. In the end, the completion time of an instruction will be dominated by the time spending on swap out/in a data to/from the hard disk. That is to say, the completion time is in an order of magnitude 10^{-3}. The performance of the computer is definitely deteriorated. Clearly, this solution is not a good option. An alternative solution is to swap the data to a RAM. The RAM, instead of the hard disk, is treated as another temporary working space for the processor.

4 Signals To/From the Processor

In simple words, an instruction consists of two parts. The first part is called the action, like MOV and ADD. The second part specifies the locations of the data. It could be a register location in the processor. It could be the location of a memory space in the memory.

Thus, a processor would need to have *instruction* signal. This signal is fed to the processor. Once a processor needs to move data to the memory, the *memory address* is thus generated and the data is moved to the *data* register. Moreover, a *control signal* (i.e. R/W) is needed to be sent out. If the control signal is 'Write', the memory address and the data are the signals to be sent out. If the control signal is 'Read', the memory address is sent out and the processor is ready to receive the data from the memory.

In principle, a processor would need to have some metal pins for receiving the instruction. These pins are connected to the 'Control Unit' in the processor. Besides, a processor would need to have some metal pins for memory address, the data and the control signals. At the same time, a memory chip would need to have metal pins for memory address, data and the control signals. These pins are physically connected to the corresponding pins of the processor.

Figure 7 is the schematic diagram showing the signals transferred between a processor and a memory. Here, we assume that RAM is not used. Moreover, one should realize that the processor and the memory are also connected to a 'Clock' circuit. This circuit generates the clock signal to all the devices in a computer. It ultimate purpose is to synchronize all devices.

Figure 8 shows a schematic diagram of a simple computer architecture. Note that the processor can interact with the ROM, the RAM, the memory and other devices like network communication card and the I/O devices. The interaction between the RAM and the memory is controlled by a system called *direct memory access*. The memory/RAM and other devices have no

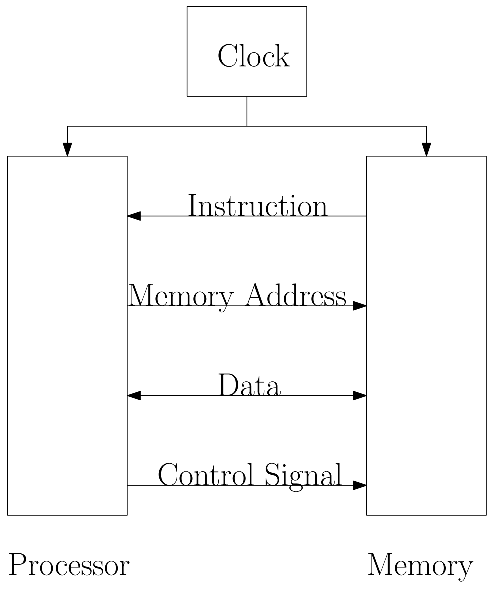


Figure 7: Types of signals transferred between a processor and a memory.

direct interaction. The processor plays a role as a centralized coordinator.

5 CISC and RISC

The set of instructions a processor can perform is called the *instruction set*. A processor which is used for general purpose is called a general purpose processor. The number of instructions in its instruction set is usually huge. This kind of processor is called the complex instruction set computing (CISC) processor. All the processors that you can find in notebooks and desktop PC are belongs to this kind of processor.

On the other hand, there are special purpose processors. For instance, some processors are designed for performing mathematical computation. Thus, the number of instructions the processor needs to perform is much smaller than the CISC processor. This kind of processor is called the reduced instruction set computing (RISC) processor. The CPU that you can find in iPhone and the processors that you find in video cards are belongs to this type. Moreover, graphical processing unit (GPU), the processor which is used in almost all video cards, is a RISC processor.

In general, the average processing time a CISC processor to perform an

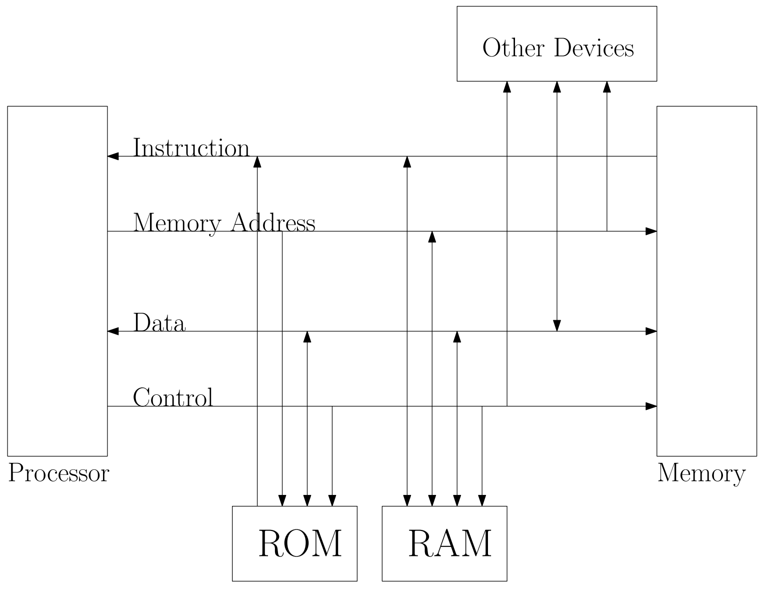


Figure 8: A simple computer architecture.

instruction is longer than a RISC processor. The circuit complexity of a CISC processor is higher than a RISC processor.

6 CPU

Central processing unit (CPU) is a digital system which is able to perform all logic and arithmetic operations. To understand how it works, one can refer to the circuits as shown in Figure 1, Figure 5 and Figure 9.

6.1 Simple ADD/SUB system

As observed from Figure 9, the signal pin on the right side controls the operation of the circuit. In other words, it controls operation of the circuit on the inputs. Thus, one can imagine a very complex circuit which consists of logic circuits handling all the arithmetic operations and other circuits handling logic operations on the inputs. We could thus control the operation of this complex circuit by sending *control signals*. But this time, the number of signal pins will be many more than one.

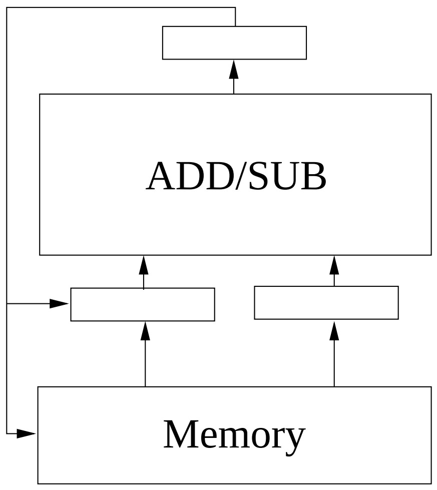


Figure 9: A system consists of arithmetic unit and memory. The control unit is not shown in the diagram.

If moreover there are (electronic) components which can be built for storage (see the blank rectangular blocks in Figure 9), we can connect them to the inputs and the outputs. Let say the temporary storage for inputs are called **IA** and **IB** and the output is called **OUT**. Precisely, it is called **register**. By using the same components, it is for sure that memory can be built. Suppose we have a memory with 16 units. Each unit consists of 4 bits. Let say the memory units are **M1**, **M2**, **M3**, $\cdots$, **M16**.

6.2 Program

Now, suppose we have a problem to find out the subtotal of **M1**, **M2**, **M3** and then save the answer on **M4**. We use the term **ADD** as the "name" for the "control signals" (binary signals) for controlling the circuit to perform addition. The following steps can help to do this task.

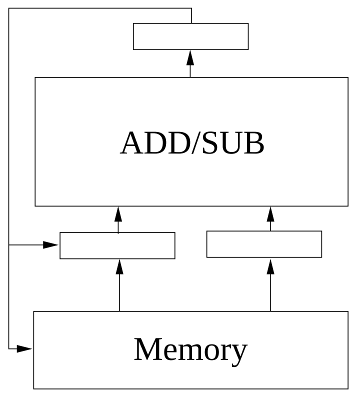
```
----------------------
A sample program

M4 = M1 + M2 + M3
----------------------
MOV IA M1
MOV IB M2
ADD IA IB
MOV IA OUT
MOV IB M3
ADD IA IB
MOV M4 OUT
----------------------
```

In the above example, the signal **MOV** is designed for signal-flow from the memory to register, and from register to register. The above steps for adding three numbers, while simple, are called instructions. These seven instructions constitute the program. *A program is a sequence of instructions.* **MOV** and **ADD** are called commands. Since this program describes what the ABB/SUB circuit and the memory should do, it is the so-called **low-level program**. The corresponding binary code is called **machine code**. The software translating the program to machine code is called **compiler**.

7 Artificial CPU (JS2019)

Below is a simple circuit. It consists of a **memory** with 16 memory spaces (from M1 to M16), an **ALU block**, 2 **input registers** (IA and IB) and one **output register** (OUT). M1 to M16, IA, IB and OUT are all 8 bits long. Numbers are represented in fixed-point *2's compliment* format.



7.1 List of Commands

Eleven commands (MOV, ADD, SUB, MUL, DIV, CMP, SHL, SHR, DEF, MSK and IF) are provided for instructing the above circuit. The syntax and the descriptions of these commands are depicted in Table 1.

Table 2: Commands for using the CPU.

Syntax	Description
MOV X Y	Copy the content of Y to X
ADD X Y	$OUT = X + Y$.
SUB X Y	$OUT = X - Y$.
MUL X Y	$OUT = X \times Y$.
DIV X Y	$OUT = X/Y$.
CMP X Y	$OUT = b_1b_2b_3b_4b_5b_6b_7b_8$.
	$b_i = 0$ if $X_i = Y_i$.
	$b_i = 1$ if $X_i \neq Y_i$.
SHL X Y	OUT is the content of X
	shifting left Y bits.
SHR X Y	OUT is the content of X
	shifting right Y bits.
DEF X N	Define X as the number N.
MSK X M	Mask the value of X by M.
IF ELSE	Condition statement.

A command is also called an instruction. The set of instruction available for instructing a CPU is called instruction set. For a general purpose CPU, like Intel CORE i5, the total number of instructions is large. So, this type of processors is called complex instruction set processor. For some special purpose processors, like GPU and digital signal processors, they are used for specialized purposes, matrix multiplication for instance. Only a small set of instructions for arithmetic operations is enough. This type of processors is called reduced instruction set processor.

7.2 Command Descriptions

1. For the "CMP" command, if $X = 0110$ and $Y = 1101$, $OUT = 1011$.

2. For "SHL" and "SHR" commands, the content of Y can only be one of the following.

Y	Meaning
10000000	(Shift 7 bits)
01000000	(Shift 6 bits)
00100000	(Shift 5 bits)
00010000	(Shift 4 bits)
00001000	(Shift 3 bits)
00000100	(Shift 2 bits)
00000010	(Shift 1 bits)
00000001	(No shift)

   For example, if

$$X = 00011000, Y = 00000100,$$

   the OUT of "SHL X Y" is 01100000 and the OUT of "SHR X Y" is 00000110.

3. For the "DEF" command, N must be a number in *decimal* form. X can only be a memory location. "DEF" command is not applicable for assigning values to a register. It is used to assign a value to a memory location. For example, "DEF M1 12" means that memory location $M1$ will be assigned with a value 12. Therefore, $M1 = 00001100$.

4. For the "MSK" command, it is used for masking a register (either IA or IB) by the mask M (in binary). The mask must be 8 bits long.

   Suppose that the content of IA and M are defined as follows :

$$IA = 01001001, M = 11110000.$$

   Then, the output OUT will be "01000000". The last four bits are masked. Here is an example.

```
------------------------
DEF M1 45
MOV IA M1
MSK IA 00001111
MOV M2 OUT
------------------------
```

Initially, $M1$ is assigned with value 45. In binary form, the content reads "00101101". Thus, the output OUT is "00001101".

5. The "IF-ELSE" command is an advanced level command. It is for conditional statement. Once it is executed, the CPU will performs multiple steps in order to make it works. You do not need to know the detail how it works. In term of its usage, it is simple. Here is an example.

```
------------------------
DEF M1 1
DEF M2 2
DEF M3 1

MOV IA M1
IF IA == 0
   MOV IA M2
   MOV IB M3
   ADD IA IB
   MOV M4 OUT
ELSE
   MOV IA M1
   MOV IB M2
   ADD IA IB
   MOV M4 OUT
ENDIF
------------------------
```

Command "IF" checks if the content of IA is identical to "0". If it is, it will perform $M2 + M3$ and output the result to $M4$. Otherwise, it will perform $M1 + M2$ and output the result to $M4$.

```
------------------------
DEF M1 1
DEF M2 2
DEF M3 1

MOV IA M1
```

```
IF IA == 0
   MOV IA M2
   MOV IB M3
   ADD IA IB
   MOV M4 OUT
ENDIF
------------------------
```

In this example, the CPU performs $M2 + M3$ only if IA is zero. Otherwise, it performs nothing.

6. For the "IF-ELSE" command, the following conditions are allowed for you to define. Here NUM must be stated in decimal form but not in binary.

```
------------------------
IA == NUM
IA > NUM
IA >= NUM
IA < NUM
IA <= NUM
------------------------
```

8 Computer Organization

Generally speaking, a computer is a huge logic circuit. For clarity, this huge logic circuit is partitioned into different sub-systems namely central processing unit, memory, input/output device, network communication device, see also Figure 8. To manage the usage of these hardware units, an operating system is needed.

8.1 Hardware

In a computer, there are five important hardware components, namely (i) the central processing unit (CPU) or processor in short, (ii) the memory, (iii) the input device, (iv) the output device and (v) the network communication device. Table 3 lists some of these devices that can be found in notebook computers.

Table 3: Common devices in a notebook computer.

Type of devices	Exemplar devices
Processor	Intel CPU, AMD CPU, Nvidia GPU
Memory	ROM, RAM, Hard disk, Solid state drive (SSD)
	USB flash memory, CD reader, External hard disk
Input	Keyboard, Mouse, Microphone, Digital camera
	Mouse pad, Touch-screen panel, Scanner
Output	Screen, Loudspeakers, Printer
Net. Comm.	WiFi card, Bluetooth card, Wired network card

8.1.1 CPU & Registers

The central processing unit (CPU) is responsible for doing arithmetic and logic operations. (CPU) would not just have an ADD/SUB system in it. A CPU consists of at least a unit called **Arithmetic/Logic Unit** (ALU), an **Instruction Decode** unit and a **Control** unit. To improve the performance of a CPU, **registers** (normally tens of Bytes) and **cache memory** (normally a few hundred KBytes) are usually embedded inside the CPU.

8.1.2 Memory

Memory (primary storage) is used for temporarily storing the data or instruction that are needed for the CPU. Usually, this primary memory is called **RAM** (random access memory). It is workable only when power is off. Once the power is off, the content in the RAM will be gone. For storing the data permanently, we need secondary storages – the hard-disk, the CD or USB flash memory. In term of memory capacity, RAM usually ranges from a few hundred mega bytes (MBytes) to a few giga bytes (GBytes). The memory capacity of a USB flash memory can now be made to 128 GBytes. Hard-disk or the latest solid state drive (SSD) has the largest capacity up to tera bytes (TBytes).

One should note that the access speed of different types of memory are quite different. By access speed, it means the time for transferring data from the memory to the CPU. Since registers and cache are embedded in the CPU, it is clear that the access speed is the fastest. RAM is mounted on the mother board. While its location is still close to the CPU, transferring data between CPU and RAM is 1000 times slower than registers. For the secondary storages like hard-disk, the access speed is even slower. Normally, it is 1000 times slower than RAM.

8.1.3 I/O devices

Input devices are those devices that can let user to enter information to the computer. Keyboard, scanner and mouse are three common input devices. Output devices are those devices that can let user to know what is happening in the computer. Monitor and loudspeaker are two common output devices.

8.1.4 NetCom devices

Strictly speaking, network communication device is not belongs to be part of a computer. Recall that the usage of the first generation electronic computer did not require network communication. But nowadays, the usage of computer is attached to Internet access. So, network communication device turns out to be an important part in a computer. Without such, the usage of a computer will be so limited.

In a computer, the network communication device is referred to the Internet communication device. It is used for making communication via the Internet. In a smartphone, there are two communication devices. One is used for making Internet communication. The other is for making telecommunication.

8.2 Operating systems

Once you have understood that a computer is just a combination of various logic circuits and devices for input/output, what you need to know more is the operating system. Operating system is a complicated program. It consists of many lines of machine code (instruction)[2]. Operating system starts running when the computer is power on and then stops when the power is off. The main purposes of the operating system is to manage the file system, the memory, the input/output devices, see Figure 10. So that, computer user can interact with the operating system via either command prompt or GUI to control the computer.

[2]Recall that machine codes are sequence of electrical signals feeding to the CPU.

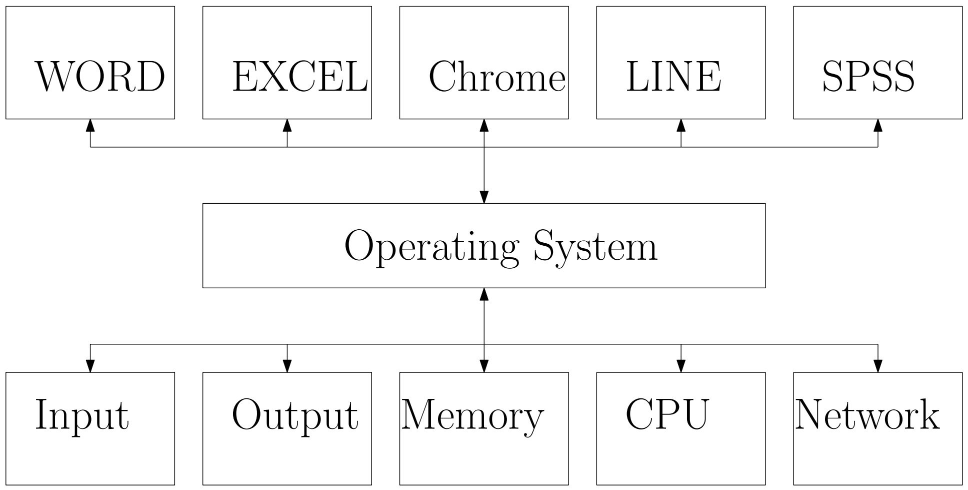


Figure 10: Operating system provides services to support the application systems.

Common operating systems are Window XP, Windows 7/8, Apple iOS for personal computers, Andorid, iOS, Window Phone 8 for smart phones. Unix and Linux are two common operating systems for network of computers.

9 Levels of Programming

In this chapter, one should realize that there are different levels of programming, namely (1) micro-programming (i.e. micro-instruction level programming), (2) machine code level programming (i.e. instruction level programming), (3) assembly level programming, (4) middle level programming and (5) high level programming. Figure 11 shows the relations amongst different levels of program. In principle, the higher the level of programming, the programming is easier. But, the functions that a program can provide are more restricted. On the other hand, the lower the level of programming, the programming is more difficult. But, the functions that a program can provide are less restricted.

One should note that a program is simply another representation of a sequence of instructions (resp. a sequence of micro-instructions). The program developed by different levels of programming could be treated as different levels of representation of the same sequence of instructions (resp. micro-

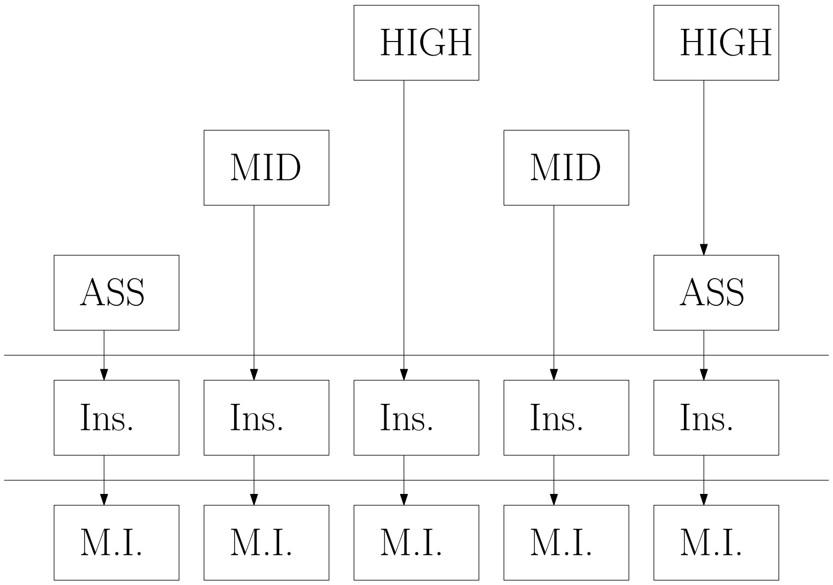


Figure 11: Level of programming or representation.

instructions).

9.1 Micro-programming

The lowest level of program is the micro-instructions. It specifies the sequence of electrical signals to be generated by the control unit to control the data flow inside a processor and the data flow amongst the processor, the memory and other hardware devices in a computer. This level of programming is called the **micro-programming**.

9.2 Machine code level programming

The next level of program is the instruction. The set of instructions is provided by the processor manufacturers based upon the architecture design of a processor. Clearly, each instruction is a set of electrical signal feeding to the processor. It is also called a machine code. It is in binary formate, not easily to be understood and not user friendly for programmer. Nevertheless, debugging such program is not an easy task. This level of programming appeared shortly in the period of the first generation electronic computers.

9.3 Assembly level programming

Sooner, an English-like programming language was introduced. Writing a program of machine codes could thus be replaced by a program of English-like programming codes. The programmer could easily debug the program if there is any syntax or logical error. Once an error-free English-like program had been ready, a human compiler could thus translate the program line by line to a program of machine code.

Thus, each processor manufacturer would develop a compiler and a programming language called assembly language. Assembly language is a English-like programming language. Each line of code refers to one machine code.

For some mechanical systems requiring real-time digital control, their controllers are normally designed with a specialized processor and the corresponding assembly language.

9.4 Middle level programming

While assembly language is more user-friendly than the machine code, it is still too difficult for a programmer to develop a program to perform a task like networking. Thus, other programming languages have been developed for handling such tasks. Some authors called them middle level programming languages. C language and Java are two examples. These programming languages consist of function calls to interact the Internet and the operating systems. Each program of this type will first be translated to assembly language and then to machine codes.

These middle level programming languages are good for developing programs for scientific researches which involve a lot of mathematical computation. Moreover, some of these programming languages include instructions for networking applications. They are applied to develop servers, like web servers and email servers, and other application software systems.

9.5 High level programming

For some applications for solving database access and managing information, these middle level programming languages would be too low level for programming. Thus, some higher level programming languages have been developed. Structural query language (SQL) is a high level programming language developed for database access and data manipulation. The programming languages provided by SPSS, SAS, Matlab and Mathematica are developed for mathematical calculations.

10 Register and Memory

Here, all the registers in the processors as shown in Figure 1 and Figure 5 are one-bit registers. In fact, the size of a register in a processor is usually a multiplication of 8 bits, like 8 bits, 16 bits, 32 bits and so on. Therefore, the actual number of logical gates and the switches is 8 times, 16 times and even 32 times higher.

Similar, each memory location (i.e. memory address) is not referred to a one-bit memory space. In reality, each memory location (i.e. memory address) is referred to an 8-bit memory space (i.e. one byte memory space). Hence, the number of switches in the memory is 8 times higher. It is a de facto standard. No one can change.

This size is good enough for storing a character defined in the ASCII table. To store a Asian language character, two *consecutive* memory locations are required. Thus, to store a 16-bit number to the main memory, two *consecutive* memory locations are required. To store a 32-bit floating point number, four *consecutive* memory locations are required.

One can imagine that multiple cycles are needed for the data transferred between the processor and the memory if the data is a number or an Asian language character.

11 A Computer versus A Firm

Basically, a computer could be analogized to a firm, see Table 4. The processor could be treated as a factory and the adminstration office of a firm. The memory could be treated as the warehouse storing the products produced from the factory. The logic gates in the processor could be treated as the tools or machines in a factory. The switches are treated as the workers controlling the flow of the semi-products. The registers are treated as the working space for putting the semi-products. The control unit in the processor could thus be treated as the management team of the firm. In this regard, an instruction to the processor could thus be treated as a service request to the firm. The instruction set is the set of services the firm can do to its customers.

11.1 An instruction versus a service

Once a service request (resp. instruction) has been received, the operations manager (resp. control unit) generates a sequence of signals (resp. micro-instructions) to command the workers (resp. switches) to control the flow of semi-products (resp. data) from one working space (resp. register) to another. The sequence of signals (resp. micro-instructions) could also be used for commanding the workers (resp. switches) to control the flow of semi-products (resp. data) from one working space (resp. register) to a specific location (resp. memory address) in the warehouse (resp. memory) and vice versa.

11.2 Control unit versus operations manager

To run a firm (resp. a computer) is basically an easy jobs if all the designs have been ready, the tools/machines (resp. logic gates) and the workers (resp. switches) perform perfectly. If the tools/machines (resp. logic gates) and the workers (resp. switches) cannot perform as they should, additional management works (resp. addition control circuits in the processor) will have to be done (resp. have to be designed). In a firm, these additional management works are thus be done by a middle management team. The team members are so-called the operations managers. They follow the pre-designed operating procedures to monitor the progress and the quality of a service. Sometimes, an operations manager will also have to design a new operating procedure for a new service or re-design the operating procedure for an existing service.

11.3 Computer designers versus firm executives

However, the most difficult task is how to design a firm (resp. a computer). Designing a firm (resp. a computer) involves three interrelated designs, namely factory design (resp. processor design), warehouse design (resp. memory device design), the operations design (resp. micro-instructions design) and the organization design (resp. architecture design). These four

Table 4: A computer versus a firm.

A Computer	A Firm
Processor	Factory
Memory	Warehouse
Logic gates	Tools/Machines
Switches	Workers
Registers	Working space for semi-products
Control unit	Operations manager
A memory address	A location in the warehouse
A memory space	A space for storing a product
Data specification	Product specification
Operating system (OS)	Executives (Top/Senior/Middle)
Set of instructions	Set of services (resp. productions)
An instruction	A service request
A micro-program	An operation procedure
Levels of programming	Levels of management
Micro-instructions design	Operating procedure design
Processor design	Factory design
Memory device design (Design for efficiently storage & retrieval of data)	Warehouse design (Design for efficiently storage & retrieval of products)
Architecture design	Organization design
Program design	Operations design
Computer designers	Executives (Top/Senior/Middle)

Either a computer or a firm, the most difficult tasks are the designs. These designs are interrelated. One design could affect the design of the others. Thus, a good design of a firm (resp. a computer) has to consider multiple aspects, i.e. co-design.

designs, especially the factory design (resp. processor design) and the warehouse design (resp. memory device design), reply very much on the selection of tools/machines (resp. logic gates) and the use of workers (resp. switches). Thus, the design of a firm (resp. a computer) is a co-design.

Now, it comes to an important question. Who is going to do these designs? For a computer, one could say that the designs would be done by the computer engineers. But, what about a firm? To design a firm, it is clearly to be done by the founders (i.e. the senior executives). Thus, the jobs an executive or a manager needs to do are (i) to design the set of services to be delivered, (ii) the design of the operations, (iii) the design of the organization, (iv) the selection of the tools/machines and (v) the recruitment of the workers. All these jobs have to be designed simultaneously. These are the actual challenges a management student has to learn.

11.4 Design principles

The principles introduced in management (resp. computer science) are simply the principles for the good designs of a firm (resp. a computer). The major challenges are the designs. A management student has to learn how to actually design. The principles behind management are just for reference. As a matter of fact, these principles could naturally be evolved once an executive has practically designed a number of operations and organizations. If a management student only learns the principles but has no any practical experience, the knowledge acquired is nothing.

12 Exercises

Question 1

Refer to the artificial CPU and its commands, what will be the content of $M4$ if the following commands are executed?

```
DEF M1 0
DEF M2 2
DEF M3 5

MOV IA M1
IF IA == 0
```

```
   MOV IA M2
   MOV IB M3
   ADD IA IB
   MOV M4 OUT
ELSE
   MOV IA M1
   MOV IB M2
   ADD IA IB
   MOV M4 OUT
ENDIF
```

Answer:

(a) 2.

(b) 7.

(c) 5.

(d) 0.

Question 2

What will be the content of $M4$ if the following program segment is executed?

```
DEF M1 16
DEF M2 22
DEF M3 10
MOV IA M1
MOV IB M2
CMP IA IB
MOV M4 OUT
MOV IA M2
MOV IB M3
CMP IA IB
MOV IA OUT
MOV IB M4
ADD IA IB
MOV M4 OUT
```

Answer:

(a) 28.

(b) 30.

(c) 32.

(d) 34.

Question 3

Refer to the artificial CPU and its commands, what will be the content of $M4$ if the following commands are executed?

```
DEF M1 0
DEF M2 2
DEF M3 5

MOV IA M1
IF IA == 0
   MOV IA M2
   SHL IA 00000100
   MOV IA OUT
   MOV IB M2
   ADD IA IB
   MOV M4 OUT
ELSE
   MOV IA M3
   SHL IA 00000100
   MOV IA OUT
   MOV IB M3
   ADD IA IB
   MOV M4 OUT
ENDIF
```

Answer:

(a) 4.

(b) 6.

(c) 8.

(d) 10.

Question 4

Three numbers have been stored in M1, M2 and M3. Which of the following program segments can correctly give the output of the following formulae?

$$M4 = M1 + M2 \times M3.$$

Answer:

(a)
```
----------------
MOV IA M1
MOV IB M2
MUL IA IB
MOV IA OUT
MOV IB M3
ADD IA IB
MOV M4 OUT
----------------
```

(b)
```
----------------
MOV IA M1
MOV IB M2
ADD IA IB
MOV IA OUT
MOV IB M3
MUL IA IB
MOV M4 OUT
----------------
```

(c)
```
----------------
MOV IA M2
MOV IB M3
MUL IA IB
MOV IA OUT
MOV IB M1
ADD IA IB
MOV M4 OUT
----------------
```

(d)
```
----------------
MOV IA M2
MOV IB M3
ADD IA IB
MOV IA OUT
MOV IB M1
MUL IA IB
MOV M4 OUT
----------------
```

Question 5

Given that there are five memories M1, M2, M3, M4 and M5. Here is the program segment to instruct the circuit.

```
----------------
MOV IA M1
MOV IB M2
MUL IA IB
MOV M5 OUT
MOV IA M3
MOV IB M4
MUL IA IB
MOV IA OUT
MOV IB M5
ADD IA IB
MOV M5 OUT
----------------
```

Which of the following mathematical equation is identical to the operation of the following program segment?
Answer:

(a) $M5 = M1 + M2 \times M3 + M4$.

(b) $M5 = (M1 + M2) \times M3 + M4$

(c) $M5 = M1 \times (M2 + M3) \times M4$.

(d) $M5 = M1 \times M2 + M3 \times M4$.

Question 6

Given that there are five memories M1, M2, M3, M4 and M5. Here is the program segment to instruct the circuit.

```
----------------
MOV IA M1
MOV IB M2
MUL IA IB
MOV IA OUT
MOV IB M3
MUL IA IB
MOV IA OUT
MOV IB M4
SUB IA IB
MOV M5 OUT
----------------
```

which of the following mathematical equation is identical to the operation of the following program segment?
Answer:

(a) $M5 = M4 - M1 \times M2 \times M3$.

(b) $M5 = M4 - (M1 + M2) \times M3$

(c) $M5 = M1 \times M2 \times M3 - M4$.

(d) $M5 = (M1 + M2) \times M3 - M4$.

Question 7

If the content in $M1$ is either '1' or '0', which of the following mathematical equation is identical to the operation of the following program segment?

```
MOV IA M1
IF IA == 0
   MOV IA M2
   SHL IA 00000100
   MOV IA OUT
   MOV IB M2
   ADD IA IB
   MOV M4 OUT
ELSE
   MOV IA M3
   SHL IA 00000100
   MOV IA OUT
   MOV IB M3
   ADD IA IB
   MOV M4 OUT
ENDIF
```

Answer:

(a) $M1 \times (9 \times M2) + (1 - M1) \times (9 \times M3)$.

(b) $M1 \times (5 \times M2) + (1 - M1) \times (5 \times M3)$.

(c) $(1 - M1) \times (9 \times M2) + M1 \times (9 \times M3)$.

(d) $(1 - M1) \times (5 \times M2) + M1 \times (5 \times M3)$.

Question 8

1. What is the truth table for the following logical operation?

$$Z = (\bar{A} + B) \oplus (A + \bar{B}).$$

2. With reference to the processor as shown in Figure 5, state (i) the assembly program and (ii) the micro-instructions for the following operation.

$$M3 = (\bar{M}1 + M2) \oplus (M1 + \bar{M}2).$$

   It is assumed that $M1$, $M2$ and $M3$ are three memory locations in the main memory. Each location is a one-bit memory. Their memory addresses are 00, 01 and 10.

Question 9

1. What is the use of the control unit in a processor?

2. What is an micro-instruction and what is a micro-program (i.e. the program consists of a sequence of micro-instructions)?
3. With an aid of example, explain why different processor architectures would come up with different instruction sets?
4. What is the purpose of the 'Clock' circuit in a computer?
5. State the different levels of programming and the reasons why there are many different levels of programming.
6. Which level of programming languages consists of function calls for accessing Internet and the operating systems.
7. What is the role of an operating system from the firm level perspective?
8. State the tasks an executive has to do to a firm.
9. State the tasks an operations manager has to do to a firm.