Study CSE309

Sofiullah Iqbal Kiron

9 December, 2022

Last compiled: December 10, 2022

L1: Introduction

The architecture of a computer can be defined as, design of task-performing part of it. Which also includes overall fundamental working principle and the internal logical structure of a computing system. So, we can say computer architecture includes:

- Instruction set architecture: attributes of computing system as seen by assembly language programmer. It also includes:
  - Data storage(where is data located?)
  - Addressing mode(how data is accessed?)
  - Exceptional conditions(what happens if something goes wrong?)
- Machine architecture: is the view that is seen by the logic designer. It includes:
  - Capabilities and performance characteristics of functional units.
  - Ways in which the components are interconnected.

  - How information flows between components.

In order to design good instruction set, it is important to understand how the architecture might be implemented. A computer consist of 3 main functional units:

1. I/O device
   - Input
   - Output
2. Memory
3. Processor
   - Arithmetic and Logic unit(ALU).
   - Control unit.

So, there is 5 main components inside functional unit.

L3: Basic Operational Concept

Previous Concept

Each activity in the computer is governed by instructions or a set of instruction. So, to perform a specific task, corresponding program(set of instruction) for that task first brought into processor from memory. Then, the data needed for this operation also brought from memory to processor. After all, the operation is executed.

An example: 1

Let **Add LocA,** R_0. The steps required to perform this instruction

1. Fetch the instruction from memory to processor.
2. Fetch the corresponding operand into processor located at location A.
3. The operand is then added to te content of R_0.

4. Store the result inside R_0.

This instruction needs to perform two instruction, one memory access operation and another, ALU operation.
To reduce the time needed and optimize/improve the performance this operation can be done separately

Memory access operation: **Load LocA,** R_1

(a) Fetch the instruction from memory to processor.
(b) Fetch the operand to R_1 which is located at location A.

ALU operation **Add** R_1**,** R_0

(a) Content of R_1 is added to R_0.
(b) After addition, store the result inside R_0.

Beside the ALU and Control Unit, processor has some registers which has some specific purpose:

IR: Instruction Register, holds the instruction currently being executed. Each of such instruction is executed under a command which is timing signal from control unit. So, the result of that instruction in IR is available in control circuit because timing signal controls all the components which are necessary for execution of that instruction.

PC: Program Counter, keeps track of the execution of a program. That is, during execution of any instruction, it contains address of the next instruction to be executed. So, the program counter actually points to the next instruction. During execution of current instruction, program counter being automatically updated/incremented.

$R_0 \ldots R_{n-1}$: General Purpose Registers, contains intermediate result during execution of an instruction.

MAR & MDR: Memory Access Register and Memory Data Register, facilitate communication with memory unit. MAR contains address of the data to be accessed from memory. MDR contains data that is to be read into or written out from memory.

Operating steps for executing a program:

1. Program get into computer through input unit.
2. Then the program resides into memory.
3. For execution, program passed to the processing unit.
4. To start the execution, PC must point to the first instruction.
5. Contents of PC copied to MAR.
6. A read signal is sent to the memory then.
7. Read content of the memory where MAR pointing to, load the content into MDR. In one such read cycle, only one word can be read. Here, the addressed word is 1^{st} instruction of the program.
8. Content of the MDR now passed to IR.
9. After entering into IR, instruction is ready to be decode and execute.
10. If operation involves arithmetic computation then operands must be fetched into ALU which can be taken through input unit or load inside any general purpose register or read directly from memory which is done by previous read cycle.
11. ALU performs the operation.
12. If the result need to be store in memory, it is passed to MDR.
13. MAR is loaded with the memory address where the data to be written, a write cycle is initiated.
14. At any point of execution, PC is automatically updated so that it can point to the next instruction to be executed.
15. The normal execution is done in the above way.
16. If anu exceptional condition occurs during the execution of that program, the execution is interrupted.
17. Interruption is a signal generated by any input/output device requesting a service to the processor.

18. Then processor listen to that request, save the current internal states which includes content of PC, general purpose register contents and some control information.
19. Save this current status inside memory.
20. Processor provides the requested service by executing an appropriate interrupt service routine.
21. After providing interrupt service, saved state of processor is restored.
22. Execution of previous incomplete program begins.

L6: Organization of Processor

Single Bus Organization

6 **General purpose registers:** $R_0 \ldots R_{n-1}$. The number and use of this general purpose registers vary from processor to processor. Normally used by programmers for any general purpose i.e. storing result of any instructor.

7 **Special purpose registers:** index registers or status pointer are kept for special purposes.

8 **Y, Z & TEMP:** these registers are transparent to the programmer. A programmer never concerned about them or these registers are never mentioned in any instruction. During the execution of any instruction, processor uses these registers for temporary storage. They are never used to store the data generated by one instruction for later use by another instruction.

9 **MUX:** A 2:1 mux, the output of this mux provides one input to the ALU. If constant value 4 is selected then ALU receives 4 in its one input and if Y is selected the ALU receives contents of the Y as its one input. The constant value 4 is normally selected for increment the content of PC.

10 **ALU(Arithmetic Logic Unit):** the main block where the arithmetic and logical operations are executed. When and instruction is fetched

from memory via MDR and loaded into IR, then IR sends it to decoder and control unit. Decoder decodes the instructions and find out which arithmetic/logic operation it consist of and where the operands of operation. According to the information found after decoding, appropriate signals are generated. Let's assume, after decoding, an addition operation found. One operand is the content of R1, another is R2 and R3 here for store the result after operation. ALU receives one operand from Y, another one is directly taken from the internal bus and then output will be passed to Z. Then from Z, the result can be passed to any register via internal processor bus. So for now, content or R1 brought into Y, content of R2 brought into internal processor bus, the select input in MUX is Y. Then, ALU takes one operand as A which is output of MUX(for now, content of Y) and another operand B taken directly from bus, add them and pass the result to bus via Z. R3 takes the result from bus later on.

Register Transfer & Organization

1. While executing an instruction, a sequence of steps are involved in transferring data from one register to another. For this, 2 control signals are used:
    (a) One control signal used to place the contents on bus from register.
    (b) Another control signal is used to load the data into register from bus.
2. The input and output of any register(say, R_i), connected to the bus via 2 switches and controlled by 2 control signals(say, R_{iin} and R_{iout} respectively).
    (a) For R_{iin}
        i. If set to 1, then data will be loaded from bus to the register R_i.
        ii. If set to 0, there is no entry into R_i.
    (b) For R_{iout}
        i. If set to 1, pass data to the bus from R_i.
        ii. If set to 0, no content/data go out. Gate locked.