Computer Concepts

Andersen Ang

First created: 2012. Last update: 2017-Feb-1

1 Computer Language

1.1 1st-Generation Language : Machine Language

- Also called *machine code*
- *Binary* : Boolean, only 1 and 0 (or $high/low$, $true/false$)
- Machine understand machine code **only**
- Different set of machine use different set of machine code, so machine code is *machine dependent*
- Machine language is $fastest$, but it is so difficult to read

1.2 2nd Generation Language : Assembly Language

- Machine code is not human readable
- Assembly language is some short form / abbreviations of some English word, for human to use
- Typical example of some assembly code
L	Load
A	Add
S	Store
- Machine never understand assembly language, so a *assembler* is needed to translate assembly language into machine language
- Assembler = Assembler + Data base, since the data base is machine dependent, thus assembly language is also machine dependent

1.3 3rd Generation Language : High Level Language

- More abstract than assembly language, for example, can handle $a = b + c - 4$;
- Thus high level language is easier to read than assembly language
- Example : C, C++, Java, FORTRAN, Dephi, PASCAL, BASIC, LIPS, PROLOG

- Machine never understand high level language, so a *compliter* or *interpreter* is needed
- The speed of high level language is not as fast as machine code
- Once you have the complier or the interpreter, the language works in any machine, thus high level language is *machine independent*

1.4 4th Generation Language

- Even more easy to use than high level language, the languae is more human like
- Example : Graphical language, query language, natural languae
- Natural Language is somewhat like those scene in the movie : the computer can understand the wording such as "give me the report this week"
- Such language (natural language) is a research topic

2 Registers

There some specific memory in the computer that perform specific task, those specific memory are *registers*

- Accumulators : A register that hold the operand to be operated
- General purpose register : Just a register space for the programmer to use.
- Status register / Condition code register : A register that the byte inside contains the information of the status of the processor
- Index register : A register that hold the displacement of the address
- Program counter : A register that hold the address of the next byte of an instruction code to be fetched from the memory to the CPU
- Instruction register : A register that hold the op-code
- Data Address register : A register that hold the address of data
- Data register : A register that hold the operand, and this operand is to be added into the operand inside accumulator
- Stack Pointer : A register that hold the location of the stack data structure.

3 Addressing Mode

Computer is "processor" + "memory"

Thus there are lots of different methods for the processor to get data from the memory, these methods are the addressing modes.

3.1 Immediate Addressing Mode

- The byte after the opcode is the operand directly : | Opcode | Operand |
- Analogy | Take square root of | +9 |
- No need to do memory fetch, fastest

3.2 Direct Addressing Mode

- The byte after the opcode is the address of the operan | Opcode | Address of the operand |
- Analogy | Read the content of | the box in room 301 |
- Need to do 1 memory fetch
- Effective Address = the address

3.3 Index Addressing Mode

- The byte after the opcode is the displacement to the index register | Opcode | Displacement |
- Analogy | Go inside | the room that is 3 room after room 301 |
- Need to do 1 memory fetch
- Effective address = displacement + address stored in index register
- Index addressing is often used when accessing a block of data, using loop

3.4 Relative Addressing Mode

- The byte after the opcode is the displacement to the current programme counter | Opcode | Displacement
- Analogy | Go inside | 2 page after the page where you are reading now |
- Need to do 1 memory fetch
- Effective address = displacment + current address stored in programme counter

$-END-$