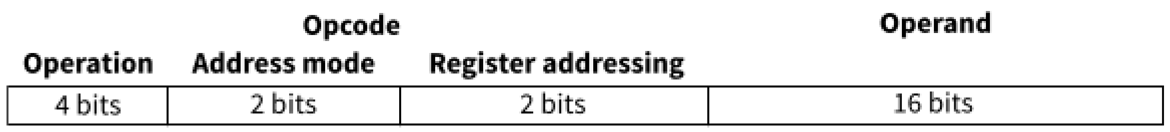
Chapter 6: Assembly language programming

Machine code instructions

- Parts:
  - Opcode: what to do
  - operand: with what



	Opcode		Operand
Operation	Address mode	Register addressing	
4 bits	2 bits	2 bits	16 bits

Assembly language and Assembler

> You could write program in machine level and it could be efficient program but not human compatible. So, we use Assembly language which uses mnemonics and characters.

- `Assembler` converts assembly language to machine code before it is executed by processor.
- Assembler also allows programmer to include:

```
- comments
- symbolic names for constants
- labels for addresses
- macros: sequence of instructions that can be reused. (like functions)
- directtives: instruction for assembler
```

- Assember does many things:
  - Remove comments
  - replace macro name with actual instructions
  - removal and storage of directives
  - actual conversion
- One pass assembler:
  - Converts complete code in one round. i.e. goes through the program once.
- Two pass assembler:
  - Useful for programs with forward reference
  - First pass: finds location of addresses for the forward reference to construct symbol table.
  - Second pass: assembler uses symbol name and lookup table to output machine code.

Addressing modes

> Addressing mode is the way of specifying value to be used by operation.

Addressing mode	Use of the operand
Immediate	The operand is the value to be used in the instruction; `SUB #48` is an example.
Direct	The operand is the address which holds the value to be used in the instruction; `ADD TOTAL` is an example.
Indirect	The operand is an address that holds the address which has the value to be used in the instruction.
Indexed	The operand is an address to which must be added the value currently in the index register (IX) to get the address which holds the value to be used in the instruction.

- For immediate addressing, you can provide value in denary, binary or hexadecimal.

- #48 specifies the denary value 48
- #B00110000 specifies the binary equivalent
- #&30 specifies the hexadecimal equivalent

Assembly language instructions:

- Types:
  - Data movement: LDM, MOV, STO
  - Input and Output: IN, OUT
  - Comparisons and jumps: JMP, CMP
  - Arithmetic operations: ADD, SUB, INC, DEC
  - Shifting operations:
    - Logical: LSL, LSR
    - Arithmetic

> In logical left shift, the most significant bit is moved to the carry bit, the remaining bit are shifted left and a zero is entered for the least significant bit.

> In cyclic left shift, the bit from most significant position goes to least significant position.

> In arithmetic shift, sign is preserved and rest is like cyclic.

Flags

> Flags are bits in status register.

- Carry flag(C): set to 1 if there is carry following an addition operation.

- Negative flag(N): set to 1 if the results of a calculation gives negative value.

- Overflow flag(V): set to 1 if there is overflow.

- Zero flag(Z): set to one if result is zero.
- Others: parity, interrupt, half-carry.

$$\begin{array}{r} 10001000 \\ +\ 11000111 \\ \hline 101001111 \\ \hline \end{array}$$

Flags:
N V C Z
0 1 1 0