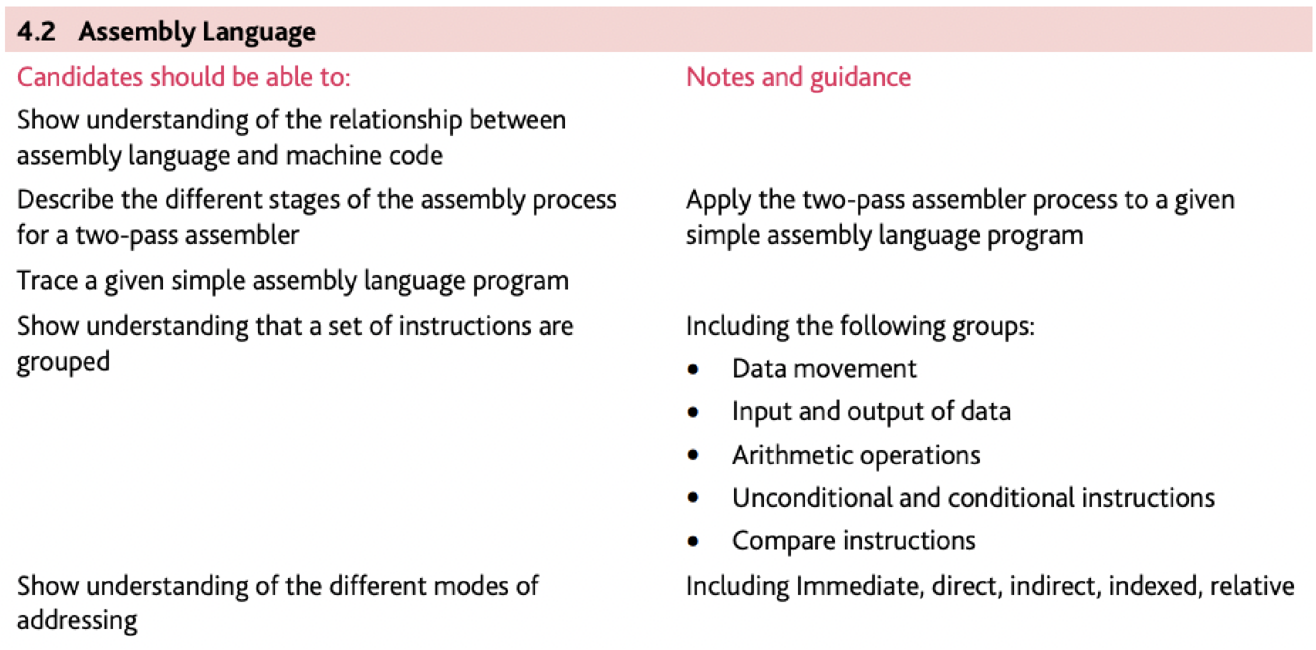
2. Assembly Language



4.2 Assembly Language

Candidates should be able to:	Notes and guidance
Show understanding of the relationship between assembly language and machine code	
Describe the different stages of the assembly process for a two-pass assembler	Apply the two-pass assembler process to a given simple assembly language program
Trace a given simple assembly language program	
Show understanding that a set of instructions are grouped	Including the following groups: • Data movement • Input and output of data • Arithmetic operations • Unconditional and conditional instructions • Compare instructions
Show understanding of the different modes of addressing	Including Immediate, direct, indirect, indexed, relative

---

- Two-pass assembler
  - Pass 1
    - Read the assembly language program one line at a time
    - Ignore anything not required, such as comments
    - Allocate a memory address for the line of code
    - Check the opcode is in the instruction set
    - Add any new labels to the symbol table with the address, if known
    - Place address of labelled instruction in the symbol table
  - Pass 2
    - Read the assembly language program one line at a time
    - Generate object code, including opcode and operand , from the symbol table generated in Pass 1
    - Save or execute the program
- Groups of instructions
  - Data movement
    - Allow data stored at one location to be copied into the accumulator

Opcode	Operand	Explanation
LDM	`#n`	Immediate addressing. Load the number to ACC

Opcode	Operand	Explanation
LDD		Direct addressing. Load the contents of the location at the given address to ACC
LDI		Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC
LDX		Indexed addressing. Form the address from + the contents of the index register. Copy the contents of this calculated address to ACC
LDR	#n	Immediate addressing. Load the number n to IX
MOV		Move the contents of the accumulator to the given register
STO		Store the contents of ACC at the given address

- Input and output of data
    - These instructions allow data to be read from the keyboard or output to the screen

Opcode	Operand	Explanation
IN		Key in a character and store its ASCII value in ACC
OUT		Output to the screen the character whose ASCII value is stored in ACC

- Arithmetic operations

    - These instructions perform simple calculations on data stored in the accumulator and store the answer in the accumulator, overwriting the original data

Opcode	Operand	Explanation
ADD		Add the contents of the given address to the ACC
ADD	#n / Bn / &n	Add the number n to the ACC
SUB		Subtract the contents of the given address from the ACC
SUB	#n / Bn / &n	Subtract the number n from the ACC

Opcode	Operand	Explanation
INC		Add 1 to the contents of the register (ACC or IX)
DEC		Subtract 1 from the contents of the register (ACC or IX)

PS: #n means that n is in denary, Bn means that n is in binary, &n means that n is in hexadecimal

- Unconditional and conditional instructions

Opcode	Operand	Explanation
JMP		Jump to the given address
JPE		Following a compare instruction, jump to if the compare was True
JPN		Following a compare instruction, jump to if the compare was False

- Compare instructions

Opcode	Operand	Explanation
CMP		Compare the contents of ACC with the contents of
CMP	#n	Compare the contents of ACC with number n
CMI		Indirect addressing. The address to be used is at the given address. Compare the contents of ACC with the contents of this second address

- Addressing modes
    - Immediate addressing
      The value of the operand is used
    - Direct addressing
      The contents of the location in the given address is used
    - Indirect addressing
      The address is the content of the given address
    - Indexed addressing
      The address is formed by adding the given address by the contents in the Index Register
    - Relative addressing
      The address is formed by adding the current address by the operand