Overview of Some Important Algorithms:

1. Assembler Module:

- Initialise address = 0;
- Traverses each text editor line one by one.
- Parse each of them.
- If the line contains any instruction increase the address by 1.
- If the instruction doesn’t matches any opcodes raise the invalid instruction error.
- If the line contains any lablel add them to the set symbols.
- If it there are more than one label with same name raise Label redefinition error.
- After all lines are parsed again traverse each text editor line one by one.
- If the line containes any instruction add its’s opcode to the array.
- If it contains jmp like instructions calculate thier address based on the set position of symbols from the previous pass.
- Return the bin array.

2. Processor Module:

- Initialise PC to the start address of the instructions.
- Search the instruction pointed by the PC register.
- If the instruction is found execute the corresponding function.
- Update the memory, flag register and other registers based on the instruction output.
- If an error occurs while executing the instruction raise runtime error.
- Increment the PC and repeat until halt is reached or count of instructions is more 5000.

3. Parser Module (Earley Algorithm):

The Earley Parsing Algorithm: an efficient top-down parsing algorithm that avoids some of the inefficiency associated with purely naive search with the same top-down strategy (cf. recursive descent parser).

- Intermediate solutions are created only once and stored in a chart (dynamic programming).
- Left-recursion problem is solved by examining the input.
- Earley is not picky about what type of grammar it accepts, i.e., it accepts arbitrary CFGs (cf. CKY).

Complete Algo at: https://en.wikipedia.org/wiki/Earley_parser