Introduction

While writing CPU program in binary is inefficiency for human, we are going to go through a very basic assembly language and compiler (written in Python) to support us in the design's verification task.

Assembly language

Command

Basing on the CPU's commands, following table is the list of supported commands and the parameter for each of them. The "NA" parameter means it is not required.

- Command list:

Command	Parameter 1	Parameter 2
ADD	RA	RB
SHR	RA	RB
SHL	RA	RB
NOT	RA	RB
AND	RA	RB
OR	RA	RB
XOR	RA	RB
CMP	RA	RB
LD	RA	RB
ST	RA	RB
DATA	RB	LD_DATA

Command	Parameter 1	Parameter 2
JMPR	RB	NA
JMP	J_ADDR	NA
JZ	J_ADDR	NA
JE	J_ADDR	NA
JA	J_ADDR	NA
JC	J_ADDR	NA
JCA	J_ADDR	NA
JCE	J_ADDR	NA
JCZ	J_ADDR	NA
JAE	J_ADDR	NA
JAZ	J_ADDR	NA
JEZ	J_ADDR	NA
JCAE	J_ADDR	NA
JCAZ	J_ADDR	NA
JCEZ	J_ADDR	NA
JAEZ	J_ADDR	NA
JCAEZ	J_ADDR	NA
CLF	NA	NA
END	NA	NA

- Comment with '#'
- One line for each command, doesn't support multiple lines for command and parameter(s)
- The compiler will assert error and stop if the number of parameters is not expected, or the command is not in the list.

Supported value for parameters LD_DATA & J_ADDR

- Number: Decimal (default), binary (0b), hexa (0x)
- Label (only used for J_ADDR). See below section.

Supported value for RA & RB: R1, R2, R3, R4 (user's registers in the design)

Label

Each command is stored in an address in memory, but it is hard to calculate the address because the commands' length are different. The assembly language support to create label to mark an address of a command so we can use jump commands easily.

- Format: `<label_name>: <command>`
- Spacing between `<label_name>` and `:` is OK
- Doesn't support multiple line for label and command
- Doesn't support label that is a number
- Doesn't support multiple definition for the same label
- Support to use label before defining it.
- Example:

```
LB1 : ADD R1 R2
      JZ LB1
```

Compiler

The compiler supports text based program, options:

```
assembly.py --input <input_file>    # Program
            --output <output_file>  # Output string that represent cpu command
            [--verbose]             # Generating debug program
```

This compiler is not case sensitive, which means `ADD` is the same as `Add` or `aDD`. It is more important when considering label name, it will assert error if labels are just different by letter case.

The output is UTF-8 format that contains string of hex number, it is not the binary number that can be written directly into the hardware. Each line is for 1 memory address. This output is used for simulation purpose. Example fo the output:

```
20
5
21
5
23
1
ea
```

Debugging message

Code	Serevity	Message	Description
E1	error	Unsupported <cmd> in line <number>	The command <cmd> at line <number> is not in the supporting list
E2	error	The number of parameters <number 1> is unexpected, line <number>. Expected <number 2>	The number of of parameters in inline <number> is <number 1> while expected one is <number 2>
E3	error	The parameter <value> value is unexpected in line <number>	The parameter <value> in line <number> is not in support list. See Command
W1	warning	The address <number 1> in line <number 2> is bigger than 255	The calculated address is too big, the CPU may not suport this address
W2	warning	END command is not found	The assembler has reached the end of the file, but it can't find the END command, which may cause CPU to run forever

Debugger

There is no debugger in this document version.