General Purpose Processor

General information:

- two 16-bit general purpose registers (X and Y)
- 16-bit accumulator
- 16-bit stack pointer
- 16-bit instructions
- 16-bit word size
- Zero, Negative, Carry and Overflow flags

Halt Instruction:

Op Code	Assembly Language	Description
000000	HLT	Halt execution

Memory Instructions:

Op Code	Assembly Language	Description
000001	LDX #immediate	Load into X the value found at address "immediate"
000001	LDY #immediate	Load into Y the value found at address "immediate"
000101	LDX (Y)	Load into X the value found at the address contained by Y
000101	LDY (X)	Load into Y the value found at the address contained by X
000010	STX, #immediate	Store the value contained by X at address "immediate"
000010	STY, #immediate	Store the value contained by Y at address "immediate"
001011	STX (Y)	Store the value contained by X at the address contained by Y
001011	STY (X)	Store the value contained by Y at the address contained by X
000011	PSH R	Push the value of R on stack
000100	POP R	Pop the value from stack into R

Arithmetic Instructions (the flags are updated after each instruction):

Op Code	Assembly Language	Description
100000	ADD R	Add to ACC the value from R, store into ACC
100001	SUB R	Subtract from the ACC the value from R, store into ACC
101000	MUL R	Multiply ACC with the value from R, store into ACC
101001	LSR R	Logical right shift ACC by R bits, store into ACC
101010	LSL R	Logical left shift ACC by R bits, store into ACC

*R can be either X or Y

101011	RSR R	Rotate right shift ACC by R bits, store into ACC
101100	RSL R	Rotate left shift ACC by R bits, store into ACC
101101	DIV R	Divide ACC with the value of R, store into ACC (integer division)
101110	MOD R	Modulo ACC with the value of R, store into ACC
101111	AND R	Bitwise and between ACC and the value of R, store into ACC
110000	OR R	Bitwise or between ACC and the value of R, store into ACC
110001	XOR R	Bitwise xor between ACC and the value of R, store into ACC
110010	NOT R	Bitwise negate R, store into ACC
110011	INC R	Increment R by one, store into R
110100	DEC R	Decrement R by one, store into R
110101	CMP R	Subtract from ACC the value of R
110110	TST R	Bitwise and between ACC and the value from R
110111	MVR R	Copy the value of R into ACC
100110	MVA R	Copy the value of ACC into R
100111	MOV R, $immediate	Set the value of R to "immediate"

Branch Instructions:

000110	BRA	Branch always
000111	BRZ	Branch if the zero flag is set
001000	BRN	Branch if the negative flag is set
001001	BRC	Branch if the carry flag is set
101111	BRO	Branch if the overflow flag is set
001100	JMP	Jump to subrutine, push current PC value to stack
001101	RET	Return from subrutine, pop first element form stack into PC

Assembler:

- requires python 3.10
- supports only line comments
- provides basic syntax checking
- addresses start with # and are specified only in hexadecimal
- numbers start with $ and are specified only in hexadecimal
- the output is a file containing the binary representations of the instructions
- usage:
    - run from terminal: python3.10 pyassembler program_file [-o destination_file]
    - if the destination file is not specified a "memfile.dat" file containing the assembled program is created in the current directory
    - in order to load and run the program the assembled contents need to be copied in the "memfile.dat" from the verilog project

*R can be either X or Y