

Usage of assembler:

1. A third-party library “bitstring” is required to run the assembler, install it using pip:
`pip install bitstring`
2. usage:
`python assembler.py <assembly_code_input.asm> <machine_code_output.txt>`

Instruction Set Architecture:

Accumulator Type processor supporting Reg-Reg, Reg-Mem and Mem-Reg operation

We have 2 types of instructions: Normal-Type Instruction and Special-Type Instruction. The type is determined by the MSB-bit of the 9-bit instruction. We call the first bit a “type bit.” The type bit determines how the next 8 bits in the instruction are interpreted.

Normal-Type Instruction:

Here, the type bit is 0. The rest of the 8 bits are as follows: 4 bits are assigned for the opcode (16 opcodes), and 4 bits are assigned for the operand (16 registers). Example: 0 0000 0001, take \$r1. The underlined bit is the type bit. It indicates that this instruction is a Normal-Type Instruction, and that the next 4 bits are opcodes: 0000 is the “take” opcode. The last 4 LSB indicate a register. In this case \$r1.

Special-Type Instruction (branch instruction):

Here, the type bit is 1. The remaining 8 bits are all used to indicate an immediate type of offset range from -128~127. This type is designed to jump to the location of a specified label, thus only one operation exists in Special-Type Instructions. Add more

format:

- for the encode part, 'i' represents the offset for branching (immediate type), 'r' represents register type, 'x' undefined yet or don't care, 'o' represents opcodes
- General purpose registers are indicated with a dollar sign \$
- r16 is the extra 1 bit register for overflow detection, which does NOT count as one of the 16 registers(\$r0~\$r15)
- \$accumulator is \$r0

b0	branch on zero
description:	branch to the LABEL, specified by the rightmost 8 bits, if the accumulator has a value of 0
operation:	if \$accumulator == 0, move program counter to the LABEL location
syntax:	b0 LABEL
type:	special-type
encode:	1 iii iiiii

take	
description:	the content of the operand register is moved(taken) into the accumulator register
operation:	\$accumulator = \$reg
syntax:	take \$reg
type:	normal-type
encode:	0 0000 rrrr

put	
description:	the content of the accumulator register is move(put) into the operand register
operation:	\$reg = \$accumulator
syntax:	put \$reg
type:	normal-type
encode:	0 0001 rrrr

load	
description:	a byte is loaded into the accumulator from the address specified by operand register
operation:	$\$accumulator = MEM[\$reg]$
syntax:	load \$reg
type:	normal-type
encode:	0010 rrrr

store	
description:	the content of accumulator is stored at the address specified by operand register
operation:	$MEM[\$reg] = \$accumulator$
syntax:	store \$reg
type:	normal-type
encode:	0 0011 rrrr

xor	Bitwise exclusive or
description:	exclusive or the operand register with accumulator and store the result into accumulator
operation:	$\$accumulator = \$accumulator \wedge \$reg$
syntax:	xor \$reg
type:	normal-type
encode:	0 0100 rrrr

nand	nand gate
------	-----------

description:	combination of and and not operation
operation:	accumulator = \$accumulator nand \$reg
syntax:	nand \$reg
type:	normal-type
encode:	0 0101 rrrr

shl	shift left
description:	Shifts accumulator left by the amount specified by a register. Zeroes are shifted in.
operation:	accumulator = \$accumulator << \$reg
syntax:	shl \$reg
type:	normal-type
encode:	0 0110 rrrr

shr	shift right
description:	Shifts accumulator right by the amount specified by a register. Zeroes are shifted in.
operation:	\$accumulator = \$accumulator >> \$reg
syntax:	shr \$reg
type:	normal-type
encode:	0 0111 rrrr

lookup	look up immediate value from the lookup memory
description:	return a value that map to the key in the lookup memory
operation:	\$accumulator = table [key]

syntax:	lookup key
type:	normal-type
encode:	0 1000 xxxx

lsn	check if acc is less than register
description:	compare a register with the accumulator and put 1 in accumulator if accumulator is less than register
operation:	if \$accumulator < \$reg, \$accumulator = 1, else accumulator = 0
syntax:	lsn \$reg
type:	normal-type
encode:	0 1001 rrrr

eql	
description:	check if acc is equal to register
operation:	if \$accumulator == \$reg, \$accumulator = 1, else \$accumulator = 0
syntax:	eql \$reg
type:	normal-type
encode:	0 1010 rrrr

add	
description:	add the operand register to the accumulator
operation:	\$accumulator = \$accumulator + \$reg
syntax:	add \$reg
type:	normal-type

encode:	0 1011 rrrr
---------	-------------

sub	
description:	sub the operand register from the accumulator
operation:	\$accumulator = \$accumulator - \$reg
syntax:	sub \$reg
type:	normal-type
encode:	0 1100 rrrr

of0	
description:	set overflow bit register to zero
operation:	\$r16 = 0
syntax:	of0
type:	normal-type
encode:	0 1101 xxxx

halt	
description:	halt the program counter
operation:	PC = PC
syntax:	halt
type:	normal-type
encode:	0 1110 xxxx

tba	TBD
description:	TBD
operation:	TBD
syntax:	TBD
type:	TBD
encode:	0 1111 xxxx