BREAD

Architecture

Table of Contents

Instructions

Name	Opcode	Syntax	Description
HLT	00	HLT	Stops the clock and must be reset
MOV	01	MOV REG, REG	Moves a the second register into the first
MOV	02	MOV REG, INT	Moves the integer into the first register
MOV	03	MOV REG, [ADDR]	Moves a spot in memory addressed by the ADDR into a register
MOV	04	MOV [ADDR], REG	Moves a register into memory addressed by the ADDR
MOV	05	MOV [ADDR], INT	Moves a integer into memory addressed by the ADDR
ADD	06	ADD REG, REG	Adds the first register to the second, and outputs it to the first register
ADD	07	ADD REG, INT	Adds the register to the integer and outputs to the register
ADD	08	ADD REG, [ADDR]	Adds the register to a spot in memory addressed by the ADDR into a register
ADD	09	ADD [ADDR], INT	Subtracts a integer to a spot in memory addressed by the ADDR and outputs to that spot in memory
SUB	0A	SUB REG, REG	Subtracts the first register to the second, and outputs it to the first register
SUB	0B	SUB REG, INT	Subtracts the register to the integer and outputs to the register

SUB	0C	SUB REG, [ADDR]	Subtracts the register to a spot in memory addressed by the ADDR into a register
SUB	0D	SUB [ADDR], INT	Subtracts a integer to a spot in memory addressed by the ADDR and outputs to that spot in memory
MUL	06	MUL REG, REG	Multiplies the first register with the second, and outputs it to the first register
MUL	07	MUL REG, INT	Multiplies the register to with integer and outputs to the register
MUL	08	MUL REG, [ADDR]	Multiplies the register with a spot in memory addressed by the ADDR into a register
MUL	09	MUL [ADDR], INT	Multiplies a integer with a spot in memory addressed by the ADDR and outputs to that spot in memory
JMP	0E	JMP INT	Jumps to the memory address (addressed by the integer) and starts executing that code
JMP	0F	JMP REG	Jumps to the memory address (addressed by the register) and starts executing that code
JMP	10	JMP [ADDR]	Jumps to the memory address (addressed by a spot in memory) and starts executing that code
JPZ	11	JPZ INT	Jumps to the memory address (addressed by the integer) and starts executing that code if the last operation resulted in 0
JPZ	12	JPZ REG	Jumps to the memory address (addressed by the register) and starts executing that code if the last operation resulted in 0

JPZ	13	JPZ [ADDR]	Jumps to the memory address (addressed by a spot in memory) and starts executing that code if the last operation resulted in 0
JNZ	14	JNZ INT	Jumps to the memory address (addressed by the integer) and starts executing that code if the last operation didn't result in 0
JNZ	15	JNZ REG	Jumps to the memory address (addressed by the register) and starts executing that code if the last operation didn't result in 0
JNZ	16	JNZ [ADDR]	Jumps to the memory address (addressed by a spot in memory) and starts executing that code if the last operation didn't result in 0
POP	17	POP REG	Pops the first item off the stack and loads it into a register
POP	18	POP [ADDR]	Pops the first item off the stack and loads it into a memory spot (addressed by ADDR)
PUSH	19	PUSH REG	Pushes a register into the stack
PUSH	1A	PUSH [ADDR]	Pops a memory spot (addressed by ADDR) into the stack

---

Registers

Name	Bits	Number
GeneralPurpose0	64/32	0
GeneralPurpose1	64/32	1
GeneralPurpose2	64/32	2

GeneralPurpose3	64/32	3
InputOutput	16	4
Instruction	64/32	(Internal)

Memory

Minimum Size: 30 Address lines (1 GigaByte)
Maximum Size: 52 Address lines (4 PetaBytes)

Assembly Examples

Repeating Add

[3] = Current Number

0: ADD GP0, 1
1: MOV [3], GP0
2: JMP 0

Exponents

[10] = Factor 1, [11] = Factor 2, GP0 = Output

```
0: MUL GP0, [10]
1: ADD GP1, 1
2: MOV GP2, GP1
3: SUB GP2, [11]
4: JNZ 0
```