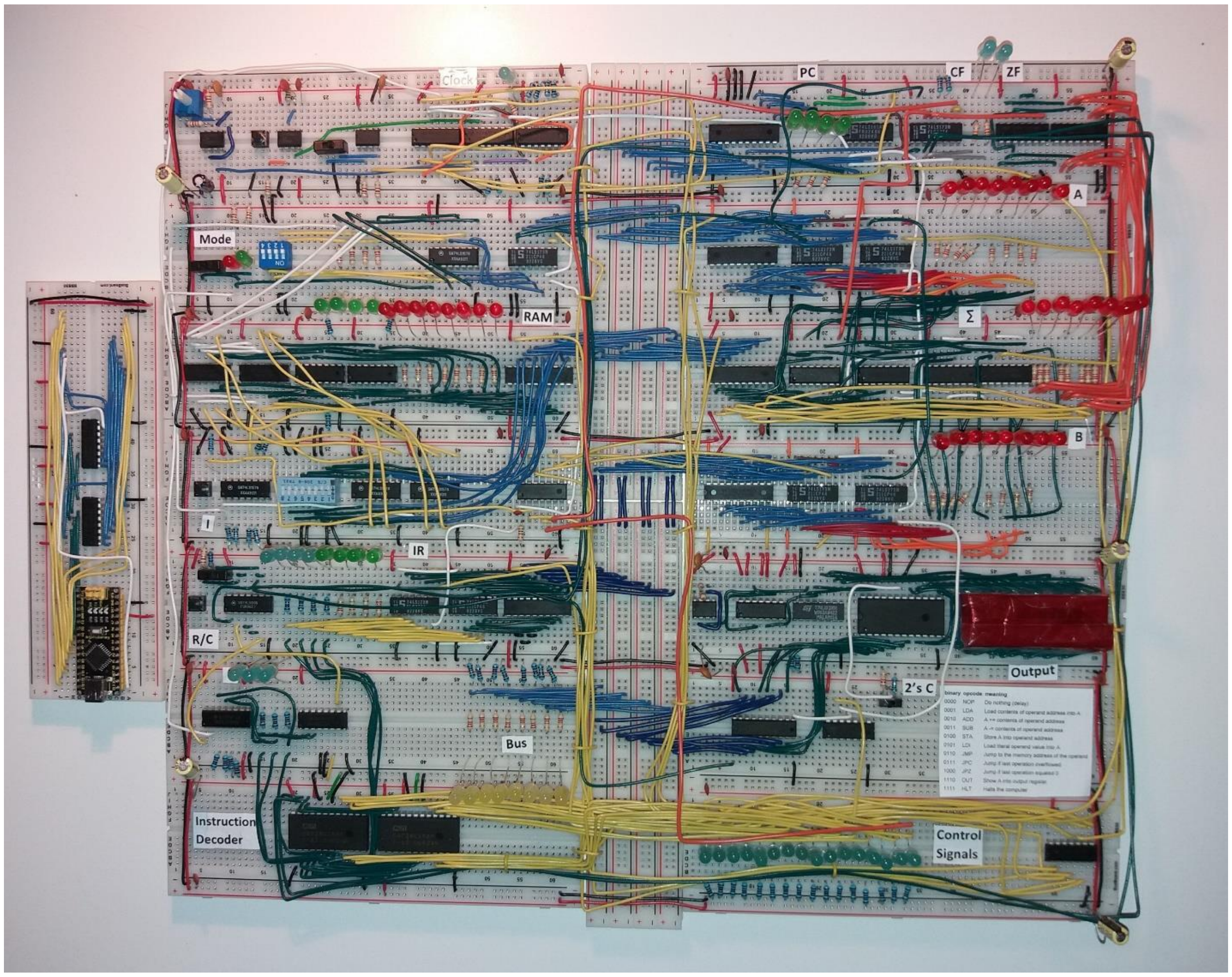
Ben Eater 8-bit computer



binary	opcode	meaning
0000	NOP	Do nothing (delay)
0001	LDA	Load contents of operand address into A
0010	ADD	A += contents of operand address
0011	SUB	A -= contents of operand address
0100	STA	Store A into operand address
0101	LDI	Load literal operand value into A
0110	JMP	Jump to the memory address of the operand
0111	JPC	Jump if last operation overflowed
1000	JPZ	Jump if last operation equaled 0
1110	OUT	Show A into output register
1111	HLT	Halts the computer

Specs

- System clock: adjustable speed from 1Hz to 300Hz, astable and monostable.
- Bus: 8-bit bus shared for data, instructions and addresses.
- Program counter: 4-bit counter.
- Memory Address: 4-bit RAM Memory Address Register
  (red led = program mode, green led = run mode).
- Memory contents: 8-bit RAM Memory contents at the memory location pointed by the Memory address register.
- Input: Input terminal to store 8-bit DIP switch value with the pushbutton at the RAM memory location pointed by the Memory Address register when in program mode.
- Resume/Clear: Inputs at run mode are determined by HLT instructions with non-zero operand that determine the RAM address where to store the input, after inputting (via switching to program mode) then the pushbutton has to be pressed with the “clear” switch already “OFF” (left) and set run mode. To completely restart the program press the pushbutton with the “clear” switch “ON” (right).
- A register: 8-bit left hand operand and register for loading and storing.
- B register: 8-bit right hand register.
- ALU: 8-bit arithmetic and logic unit that can add and subtract operands A and B and store the 9th bit carry.
- Flags register: Stores last 9th bit carry and last operation equal zero.
- Output: 8-bit register and display (555 timer, JK flip flop 2-bit digit’s place counter, 2 to 4 decoder, EEPROM decoder, 7-segment LEDs) .
- Instruction register: 8-bit instruction register with 4-bit opcode, 4-bit operand.
- CC counter: Control circuitry 3-bit counter that uses inverted system clock signal.
- Instruction decoder: 2 EEPROM that takes opcode, CC counter, Left (gnd)/Right (vcc) EEPROM, jump carry, jump zero as address signals and outputs the microcode at system low clock pulse for the next high clock pulse execution.
- Control signals: Halt (HLT), Memory address Register In (MI), RAM in (RI), RAM out (RO), Instruction register out (IO), Instruction register in (II), A register in (AI), A register out (AO), ALU out (EO), Subtract (SU), B register in (BI), Output register in (OI), Clock enable (CE), Clock out (CO), Jump (J), Flags register in (FI).

Assembly language

To program the computer set the slide switch at the Memory Address Register to light up the red led, the DIP switch at the memory address register does not require a pushbutton to update the contents, but the DIP switch for the RAM memory contents need the pushbutton at the input terminal to be pressed to save the contents. Slide the switch of the Memory Address Register to light up the green led to run the program.

binary instruction	**assembly opcode**	**has operand**	**meaning**
0000	NOP	no	Do nothing (no operation)
0001	LDA	yes	Load contents of operand address into register A
0010	ADD	yes	Load contents of operand address into register B, and set ALU to sum (default), then store result into register A
0011	SUB	yes	Load contents of operand address into register B and set ALU to subtract, then store result into register A
0100	STA	yes	Store the contents of register A into the address of the operand
0101	LDI	yes	Load immediate value of the operand into register A
0110	JMP	yes	Jump to the memory address of the operand (aka code line)
0110	JPC	yes	Jump to the memory address of the operand if last operation had a 9th bit carry
0111	JPZ	yes	Jump to the memory address of the operand if the last operation resulted in zero
1110	OUT	no	Load contents of register A into output register
1111	HLT	yes[1]	Halt the system

Download the compiler at https://github.com/skirienkopanea/8bit.

Complete documentation at skirienkopanea.github.io/8bit.

[1] To implement a runtime input use HLT with non-zero operand to store an input (thus switching back to program mode) in the memory address pointed by the operand, then switch back to run mode and hit the unstop/reset pushbutton with the “clear” switch off (left).

Sample Programs

Two runtime inputs addition

```
HLT 15
LDA 15
STA 14

HLT 15
LDA 15

ADD 14
OUT
JMP 0

. //junk
.
.
.
.
.
.

. // all runtime inputs
```

Multiply two integers

```
LDA 13
ADD 14
STA 13  //result += a

LDA 15
SUB 12
STA 15  //b--

JPZ 8   // skip loop
JMP 0   // loop

LDA 13
OUT     // output result
HLT
.       //junk

1 # b's decrement
0 # should be reset to 0 each time
7 # a
3 # b
```

Fibonacci

```
LDI 1     (part 1 of resetting the program to start with 0 and 1
STA 15    (part 2 of resetting the program)
LDI 0     (part 3 of resetting the program)

OUT       (should display 0, 1, 2, 3, 5, 8, 13, 21, 34. 55. 89, 144, 233)

JPC 0     (restart program if overflow)

ADD 15    (now we have A += last number)
STA 13    (store the updated last number in temp location)
LDA 15    (moving previous last number to second last number location)
STA 14    (part 2 of moving previous last number to second last number)
LDA 13    (part 1 of moving the new last number to last number location)
STA 15    (part 2 of moving the new last number to last number location)
LDA 14    (store A with second last number)
JMP 3     (jump to output second last number)

0         (temp value)
0         (second last number)
1         (last number)
```

Breadboard Labels and control signals

Clock Bus PC RAM I (Input) R/C (Resume/Clear) A (Left hand side operand register) B (right hand side operand register) ∑ (ALU) CF (Carry flag) ZF (Zero flag) Output 2’s C (Two’s complement) IR (Instruction register) T (Microinstruction step) Mode

(Top to bottom order in relation to their left to right order in the breadboard)

HLT	**(Halt)**
MI	**(Memory Address Register In)**
RI	**(RAM in)**
RO	**(RAM out)**
IO	**(Instruction Register out)**
II	**(Instruction Register in)**
AI	**(A register in)**
AO	**(A register out)**
EO	**(ALU out)**
SU	**(Subtract)**
BI	**(B register in)**
OI	**(Output register in)**
CE	**(Counter enable (increment))**
CO	**(Counter out)**
J	**(Jump (counter in))**
FI	**(Flags register in)**