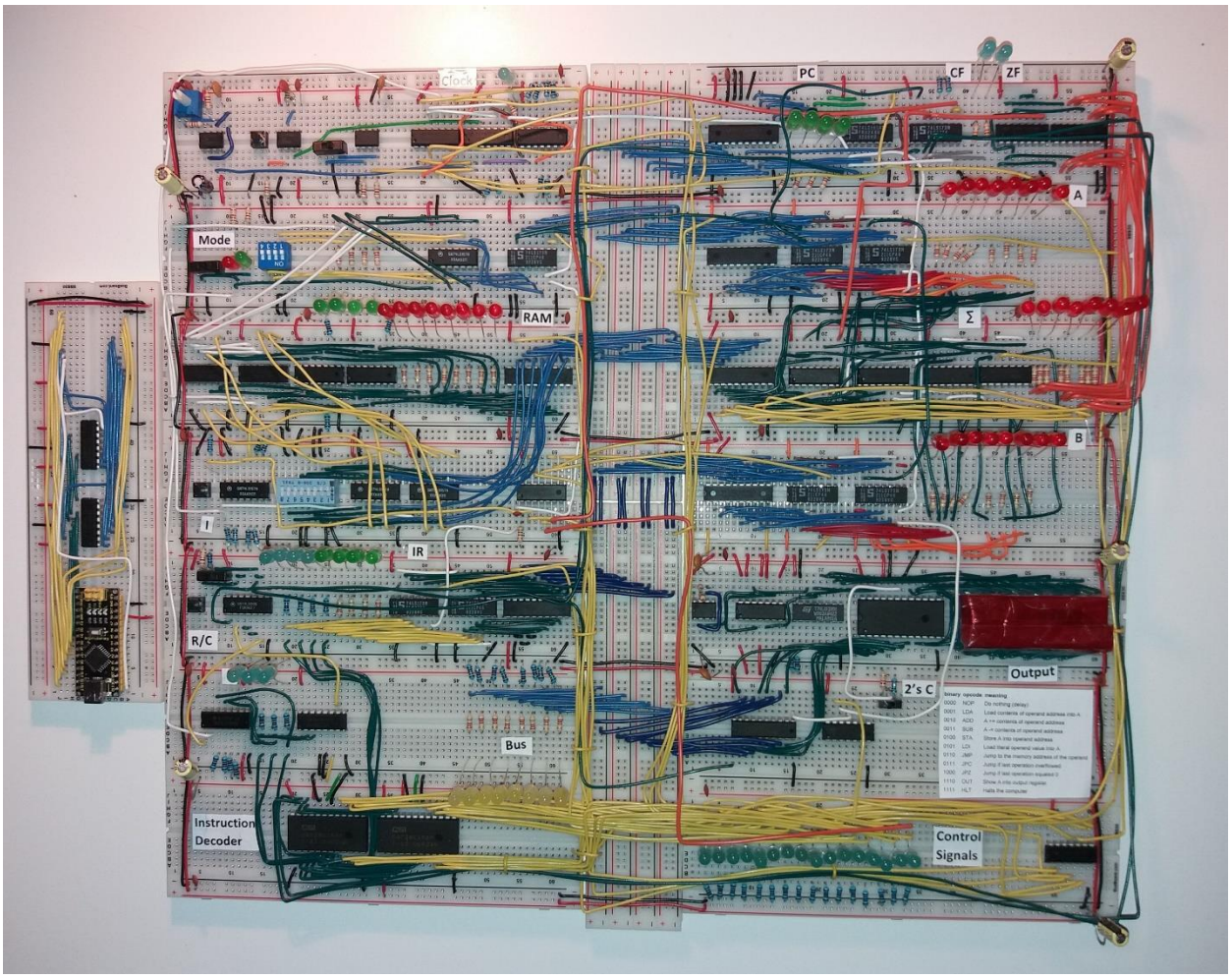
Ben Eater 8-bit computer



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 4-bit DIP switch next to the memory address register
- 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 9^{th} bit carry.
- Flags register: Stores last 9^{th} 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

```
address: contents     # assembly
0000: 1111 1111       # 0.  HLT 15
0001: 0001 1111       # 1.  LDA 15
0010: 0100 1110       # 2.  STA 14
0011: 1111 1111       # 3.  HLT 15
0100: 0001 1111       # 4.  LDA 15
0101: 0010 1110       # 5.  ADD 14
0110: 1110 0000       # 6.  OUT
0111: 0110 0000       # 7.  JMP 0
1000: 0000 0000       # 8.  0
1001: 0000 0000       # 9.  0
1010: 0000 0000       # 10. 0
1011: 0000 0000       # 11. 0
1100: 0000 0000       # 12. 0
1101: 0000 0000       # 13. 0
1110: 0000 0000       # 14. 0
1111: 0000 0000       # 15. 0 // inp
```

Multiply two integers

```
address: contents     # assembly
--------------------------------
0000: 0001 1101       # 0.  LDA 13
0001: 0010 1110       # 1.  ADD 14
0010: 0100 1101       # 2.  STA 13
0011: 0001 1111       # 3.  LDA 15
0100: 0011 1100       # 4.  SUB 12
0101: 0100 1111       # 5.  STA 15
0110: 1000 1000       # 6.  JPZ 8
0111: 0110 0000       # 7.  JMP 0
1000: 0001 1101       # 8.  LDA 13
1001: 1110 0000       # 9.  OUT
1010: 1111 0000       # 10. HLT
1011: 0000 0000       # 11. 0
1100: 0000 0001       # 12. 1
1101: 0000 0000       # 13. 0 (y)
1110: 0000 0111       # 14. 7 (a)
1111: 0000 0011       # 15. 3 (b)
// set y each time to 0
```

Fibonacci

```
address: contents     # assembly
--------------------------------
0000: 0101 0001       # 0.  LDI 1
0001: 0100 1111       # 1.  STA 15
0010: 0101 0000       # 2.  LDI 0
0011: 1110 0000       # 3.  OUT
0100: 0010 1111       # 4.  ADD 15
0101: 0111 0000       # 5.  JPC 0
0110: 0100 1101       # 6.  STA 13
0111: 0001 1111       # 7.  LDA 15
1000: 0100 1110       # 8.  STA 14
1001: 0001 1101       # 9.  LDA 13
1010: 0100 1111       # 10. STA 15
1011: 0001 1110       # 11. LDA 14
1100: 0110 0011       # 12. JMP 3
1101: 0000 0000       # 13. 0
1110: 0000 0000       # 14. 0
1111: 0000 0001       # 15. 1
```

Incrementing counter

```
address: contents     # assembly
--------------------------------
0000: 0101 0000       # 0.  LDI 0
0001: 1110 0000       # 1.  OUT
0010: 0010 1111       # 2.  ADD 15
0011: 0111 0101       # 3.  JPC 5
0100: 0110 0001       # 4.  JMP 1
0101: 0001 1111       # 5.  LDA 15
0110: 0010 1110       # 6.  ADD 14
0111: 0100 1111       # 7.  STA 15
1000: 0110 0000       # 8.  JMP 0
1001: 0000 0000       # 9.  0
1010: 0000 0000       # 10. 0
1011: 0000 0000       # 11. 0
1100: 0000 0000       # 12. 0
1101: 0000 0000       # 13. 0
1110: 0000 0001       # 14. 1
1111: 0000 0001       # 15. 1
```

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)