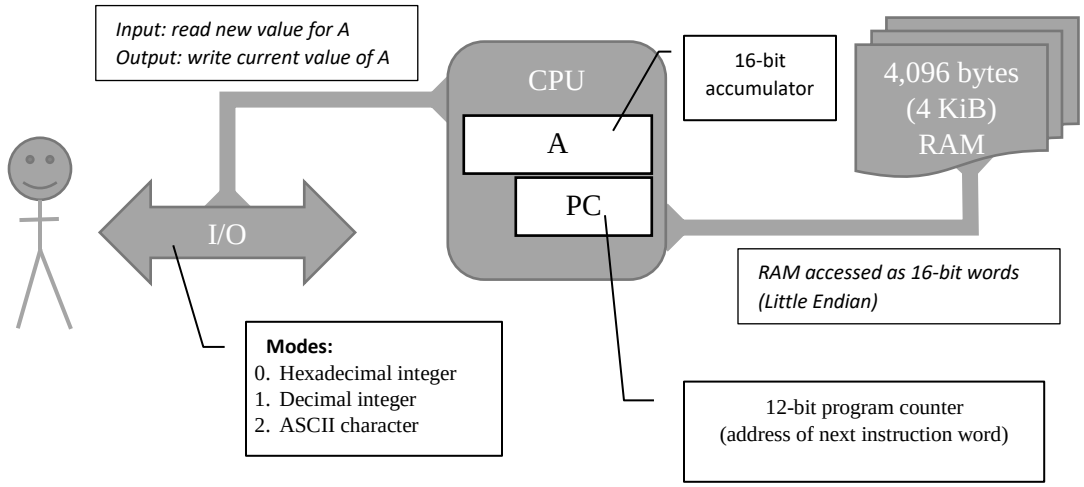
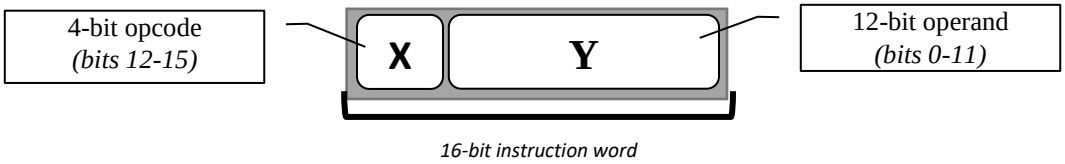
Big Boy Computer

Architecture & Organization

Component Block Diagram



Instruction Encoding



Instruction Set

Mnemonic	Opcode ("X")	Meaning	Assembly Example
`hlt`	0	*halt execution*	`hlt`
`not`	1	A = ~A;	`not`
`shl`	2	A = A << Y;	`shl 2`
`shr`	3	A = A >> Y;	`shr 8`
`inc`	4	A++;	`inc`
`dec`	5	A--;	`dec`
`jmp`	6	PC = Y;	`jmp top_of_loop`
`jaz`	7	if (A == 0) PC = Y;	`jaz escape_from_loop`
`lda`	8	A = RAM[Y];	`lda variable1`
`sta`	9	RAM[Y] = A;	`sta variable2`
`add`	10	A = A + RAM[Y];	`add sum_variable`
`and`	11	A = A & RAM[Y];	`and mask_variable`
`orr`	12	A = A | RAM[Y];	`orr mask_variable`
`xor`	13	A = A ^ RAM[Y];	`xor mask_variable`
`out`	14	write(A, mode=Y);	`out 1 ; write decimal`
`inp`	15	A = read(mode=Y);	`inp 0 ; read hex`

Example Assembly Program (Decimal Summation Program)

```
loop                    ; label: top of summation loop
      inp   1           ; read DECIMAL number from user into Acc
      jaz   stop        ; break out of loop if Acc == 0
      add   total       ; add the contents of “total” (from RAM) to Acc
      sta   total       ; store Acc into “total” (in RAM)
      out   1           ; print newly updated sum in DECIMAL form
      jmp   loop        ; repeat

stop  hlt               ; if we ever get here, halt BBC

total data  0           ; reserve space for “total” (initially 0x0000)
```

Corresponding Addresses/Machine Code Words/Disassembly

```
0x0000      0xf001      inp 1
0x0002      0x700c      jaz 0x00c
0x0004      0xa00e      add 0x00e
0x0006      0x900e      sta 0x00e
0x0008      0xe001      out 1
0x000a      0x6000      jmp 0x000
0x000c      0x0000      hlt
0x000e      0x0000      hlt         ; Note: this was the “data 0” line
```