4-bit CPU Architecture and Assembly.

Keith Allatt
January 27th, 2019

1 CPU Architecture

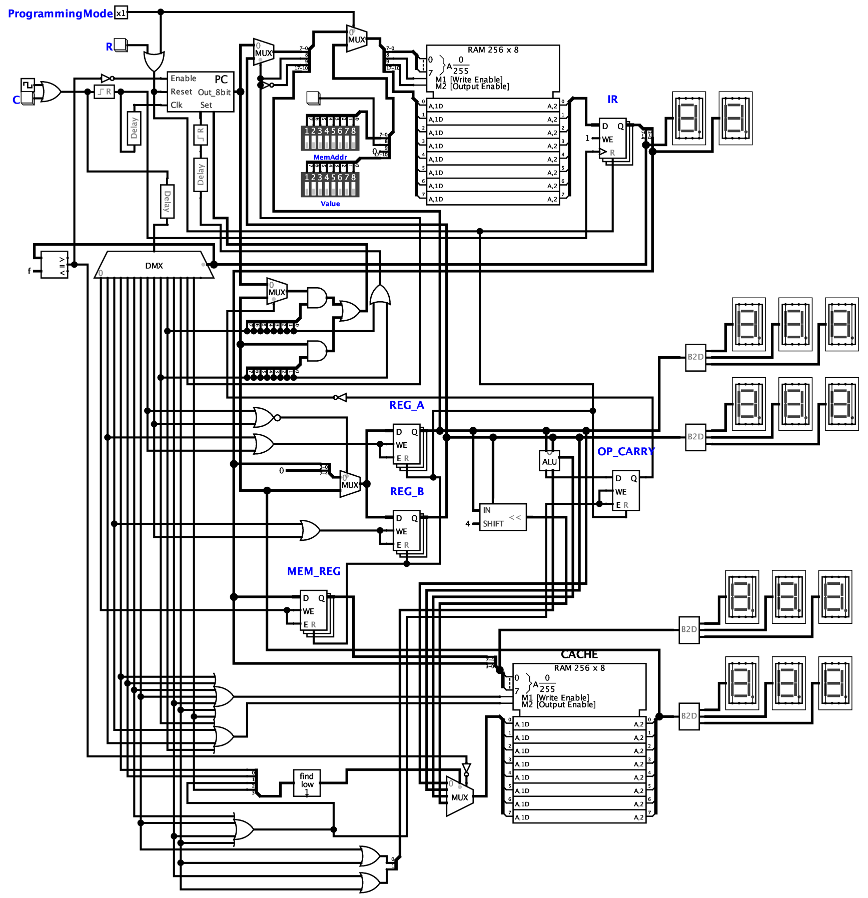

Figure 1: CPU Architecture

ALU that can compute the sum or difference between two values

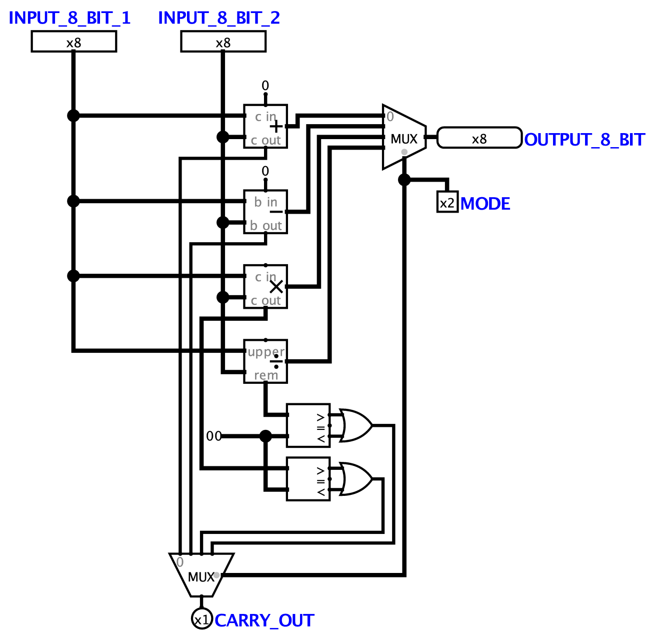

Figure 2: ALU Design

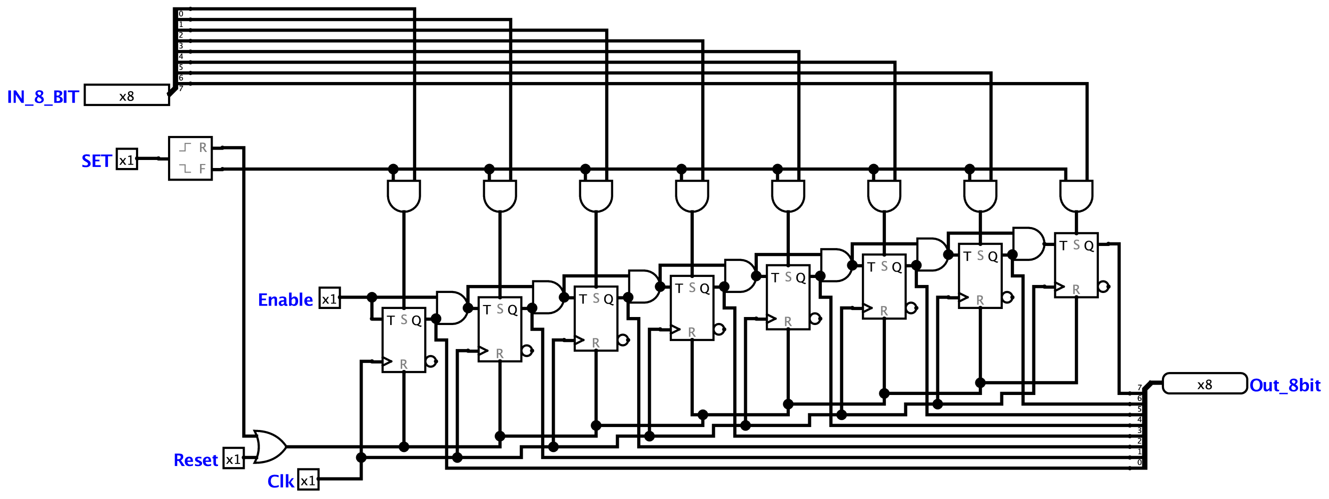

Figure 3: Program Counter Design

- Can perform addition and subtraction, multiplication and division
- Has 2 registers for immediate use
- Has 256 bytes of cache memory
- Uses a 256 byte RAM instruction list
- Programmable

2 Assembly

Each 8-bit assembly command comes in two parts, the Assembly Code, the 4 most significant bits, and the Memory Address, or Mem Addr for short, the 4 least significant bits. Sometimes, the memory address is used as a raw value, and other times it refers to a memory address in cache.

Code	Name	Function	Input	Output
0	SET MEM	Set the 4 MSBs of the mem addr	Mem Addr	Mem Reg
1	LOAD A	Load from cache into Register A	Cache	Reg A
2	LOAD B	Load from cache into Register B	Cache	Reg B
3	WRITE A	Write to cache from Register A	Reg A	Cache
4	WRITE B	Write to cache from Register B	Reg B	Cache
5	ADD A B	Write A+B to cache	Reg A, B	Cache
6	SUB A B	Write A-B to cache	Reg A, B	Cache
7	SET A	Set the value in Register A	Mem Addr	N/a
8	SET B	Set the value in Register B	Mem Addr	N/a
9	NC JUMP	Jump to line	Mem Addr	N/a
a	C JUMP	Jump if carry bit is 0, cont. when 1	Mem Addr	N/a
b	MUL A B	Write A×B to cache	Reg A, B	Cache
c	DIV A B	Write A÷B to cache	Reg A, B	Cache
d	MOD IR	Write Reg A to a IR at MA in Reg B	N/a	IR
e	LS REG	Write A << 4 + B to cache	N/a	Cache
f	HALT	Program Halts	N/a	N/a

2.1 Simple Operations and Programs

Many simple tasks don't require changing memory address pages but many do as well. Here are some simpler code snippets.

2.1.1 Clearing a page of Cache

In this context, a page of cache refers to a 16 byte chunk of memory where the 4 most significant bits remain constant. For example, a memory addresses in the 0th page would be somewhere in the range 00 to 0f, the 2nd page in the range 20 to 2f, the last page somewhere in the range f0 to ff.

Code	Name	Description
70	SET A	Set Reg A to 0
30	WRITE A	Write A, which is 0, into cache at mem. addr. 0
31	WRITE A	Write A, which is 0, into cache at mem. addr. 1
32	WRITE A	Write A, which is 0, into cache at mem. addr. 2
⋮	⋮	⋮
3e	WRITE A	Write A, which is 0, into cache at mem. addr. e
3f	WRITE A	Write A, which is 0, into cache at mem. addr. f
ff	HALT	Program is done. Cache is cleared.

2.1.2 Putting an 8-bit number into a register

For arguments sake, let us be putting an 8-bit number into register A, and let mem. addr. 0 in the cache be free. If not, replace the calls to mem. addr. 0 with the mem. addr. that's free.

Let the number being inputted be represented as XY where X and Y are hexadecimal digits.

Code	Name	Description
7X	SET A	Set Reg A to X
8Y	SET B	Set Reg B to Y
e0	LS REG	Left shift Reg A by 4 places and add B, store in mem. addr. 0
10	LOAD A	Load the value XY in mem. addr. 0 to Reg A
80	SET B	Set Reg B to 0
40	WRITE B	Write the contents of B, the value 0, to the cache at mem. addr. 0
ff	HALT	Program is done. The value XY is in Reg A

2.1.3 Switching Registers

For arguments sake, let the mem. addr.'s 0 and 1 be free.

Code	Name	Description
30	WRITE A	Write Reg A to mem. addr. 0
41	WRITE B	Write Reg B to mem. addr. 1
11	LOAD A	Load Reg A from mem. addr. 1
20	LOAD B	Load Reg B from mem. addr. 0