University of Puerto Rico - Río Piedras Campus
Department of Computer Science
CCOM-4086 Computer Architecture
Dr. Humberto Ortíz Zuazaga
Bianca I. Colón Rosado
801-13-1412

Design and implementation of a simple 16-bit CPU

We implement an 16-bit CPU up from logic gates in Logisim. Only using and gates, or gates or inverters. We used RAM for the program memory, but we built everything else up.

Our CPU will execute 16-bit instructions. With 16 registers we can let 4 bits represent the rs, rt and rd registers, and have 4 bits for the op.

This is our Control Table:

Instruction		OpCode	RegDest	Jump	Branch	MemRead	MemtoReq	MemWrite	AluSrc	RegWrite	AluCtrl	Hexadeciamal
add	rd rs rt	0000	0	0	0	0	0	0	010	1	010	002a
sub	rd rs rt	0001	0	0	0	0	0	0	010	1	110	002e
lw	rd rs rt	0010	0	0	0	1	1	0	010	1	010	032a
and	rd rs rt	0100	0	0	0	0	0	0	010	1	000	0028
or	rd rs rt	0101	0	0	0	0	0	0	010	1	001	0029
addi	rd imm 8	1000	1	0	0	0	0	0	100	1	010	104a
sw	rd rs rt	0011	0	0	0	0	0	1	010	0	010	00a2
j	imm 12	1111	0	1	0	0	0	0	100	0	010	0842
not	rd rs rt	0110	0	0	0	0	0	0	010	1	011	002b
ldhi	rd imm 8	1001	1	0	1	0	0	0	100	0	100	1444
bz	rd imm 8	1010	1	0	1	0	0	0	100	0	010	1442
bnz	rd imm 8	1011	1	0	0	0	0	0	100	0	010	1442
jal	rd imm 8	1101	1	1	0	0	0	0	100	0	010	1842
jr	rd	1110	1	1	0	0	0	0	000	0	010	1802

We use this site to calculate the hexadecimal value. From RegDest to AluCtrl.
(http://www.binaryhexconverter.com/binary-to-hex-converter)

We prove our CPU with two Programs.

- This small program count to n in 16-bit PseudoMIPS Assembly:

```
        add $1, $0, $0
        addi $1, 10          ; set n to 10
        add $2, $0, $0       ; set i to 0
loop:
        addi $2, 1
        addi $1, -1
        bnz $1, loop         ; loop is PC - 2
```

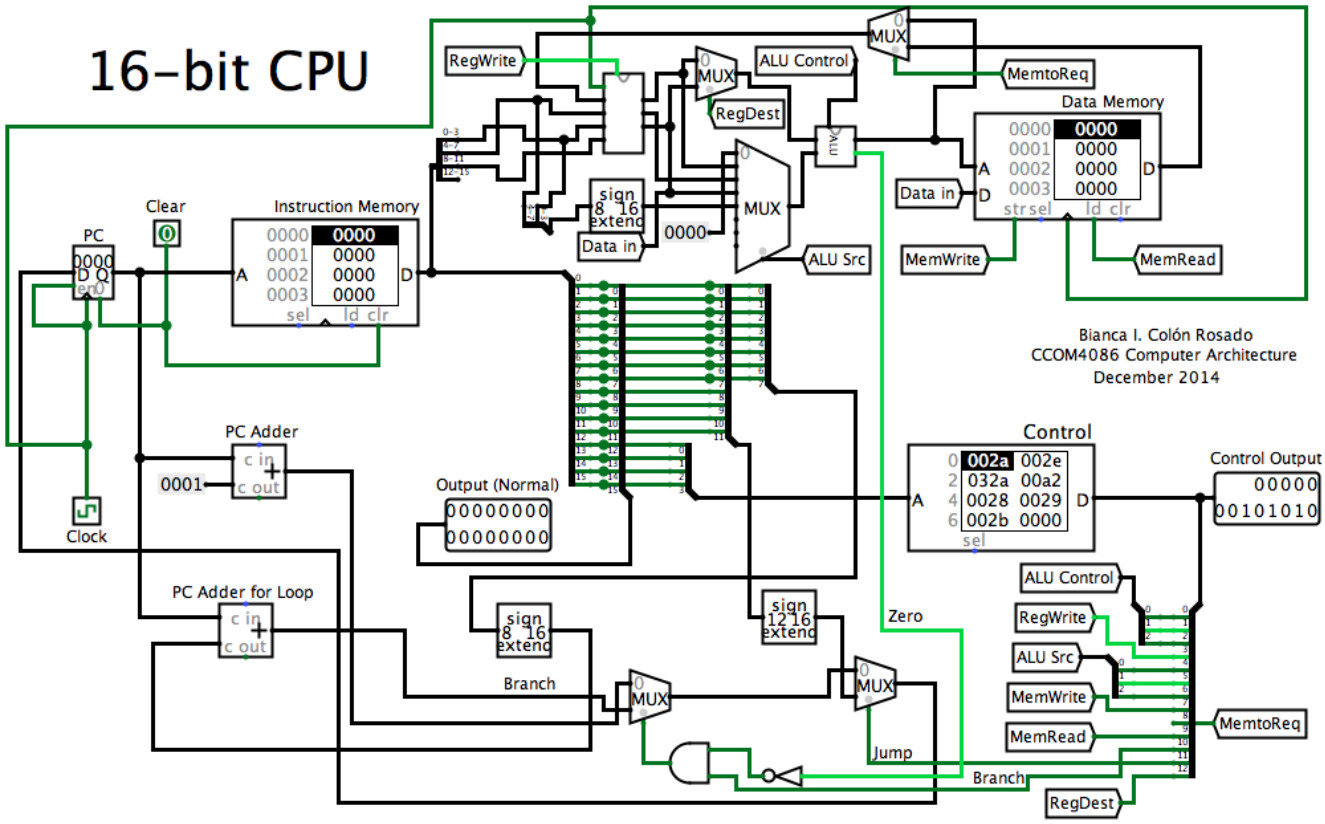
We can translate this by hand to the machine language:

binary	hex
0000 0001 0000 0000	0100
1000 0001 0000 1010	810A
0000 0010 0000 0000	0200
1000 0010 0000 0001	8201
1000 0001 1111 1111	81FF
1011 0001 1111 1110	B1FE

Loading the instruction memory with these values will allow us to simulate the execution of the program. (http://www.hpcf.upr.edu/~humberto/courses/arch2013/less-broken-cpu.html)

- The other program Compute 10 elements of Fibonacci Sequence. (https://gist.github.com/humberto-ortiz/7467201)

The design is a simplified version of the MIPS design presented in the textbook David A Patterson, John L Hennessy, *Computer Organization and Design, Fourth Edition: The Hardware/Software Interface*, 2009. ISBN 978-0-12-374493-7



A register file with 16-bit registers feeds into a simple ALU. We use multiplexers, decoders and demultiplexers to select the registers for the ALU to process. A control table drives the components based on the instruction to be executed.

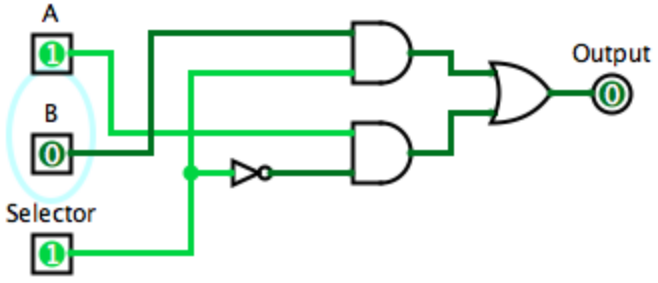
Gates:

Not = , and = , or =

Multiplexer:

The output reflect the value that has our Selector: A = 0, B= 1. The value or A = 1, and the value of B = 0. Our Selector = 1 = B, the output is 0 because that was the value of B.

A	B	Selector	Output
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



Full Adder:

Adds 3 one bit inputs, outputs result and carry.

A	B	C1	Output	C0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Our Instruction Memory and Data Memory are RAM. And our Control is ROM.

Example:

Code instruction:

addi $1, 10 ; set n to 10

4 bits representation of addi is rd imm 8

Translate the instruction to the 4 bit representation:

1000 0001 0000 1010 in hexadecimal is 810A.

Send 1000 (addi) to the control. The control send the instruction to add 0000 + 1010 (0+10), and save the result in the register 1 ($1).