Jared Wheeler
Joshua Langley
Dr. Yum
CSCE 312 - 503
Dec 4, 2016

Project Group Report

Command	Binary	Hex	Description
irmovl	0x1110 ————	0xe-	This instruction takes a register in the next four bits, and prepares to insert that number into the given register. Example: e4 20 will put the digit number 32 into register %ecx
rmmovl	0x1101 ————	0xd-	This will put the contents of a given register into a place in RAM. Example: d4 80 will place the contents of register %ecx into the memory slot 80 in the RAM memory. The first four bits are the command, the second four are the register being commanded. The program should know to grab the next byte in-order to place the value in that memory address.
mrmovl	N/A	N/A	N/A

Command	Binary	Hex	Description
addl	0x0010 ————	0x2-	Adds the contents of two registers. Example: register %ecx has the digit 32 (20 or 0010 0000) inside of it, and register %edx has the digit 3 inside of it (03 or 0000 0011). What it will do is add 32 and 3 to make 35 (23 or 0010 0011).
halt	0x0111 ————	0x7-	Basically skips the place in memory or doesnt do anything. It is looking for a starter 7-.
%eax	0x———— 0000	0x—0	The first register, represented by a 0 in RAM. All registers need one of the two implemented commands in order to use it, as the four bits will always follow the four bits of a command.
%ebx	0x———— 0100	0x—4	The second register, represented by a 4 in RAM.
%ecx	0x———— 1000	0x—8	The third register, represented by an 8 in RAM.
%edx	0x———— 1100	0x—c	The fourth register, represented by a c in RAM.

```
0x0000:                | Main:
0x0000: 30f001000000 | irmovl $1, %eax              | 30 f0 01 00
0x0006: 30f300000000 | irmovl $0, %ebx              | 30 f3 00 00
0x000c: 400300000000 | rmmovl %eax, 0(%ebx)
0x0012: 30f002000000 | irmovl $2, %eax
0x0018: 400304000000 | rmmovl %eax, 4(%ebx)
0x001e: 30f003000000 | irmovl $3, %eax
0x0024: 400308000000 | rmmovl %eax, 8(%ebx)
0x002a: 30f004000000 | irmovl $4, %eax
0x0030: 40030c000000 | rmmovl %eax, 12(%ebx)
```

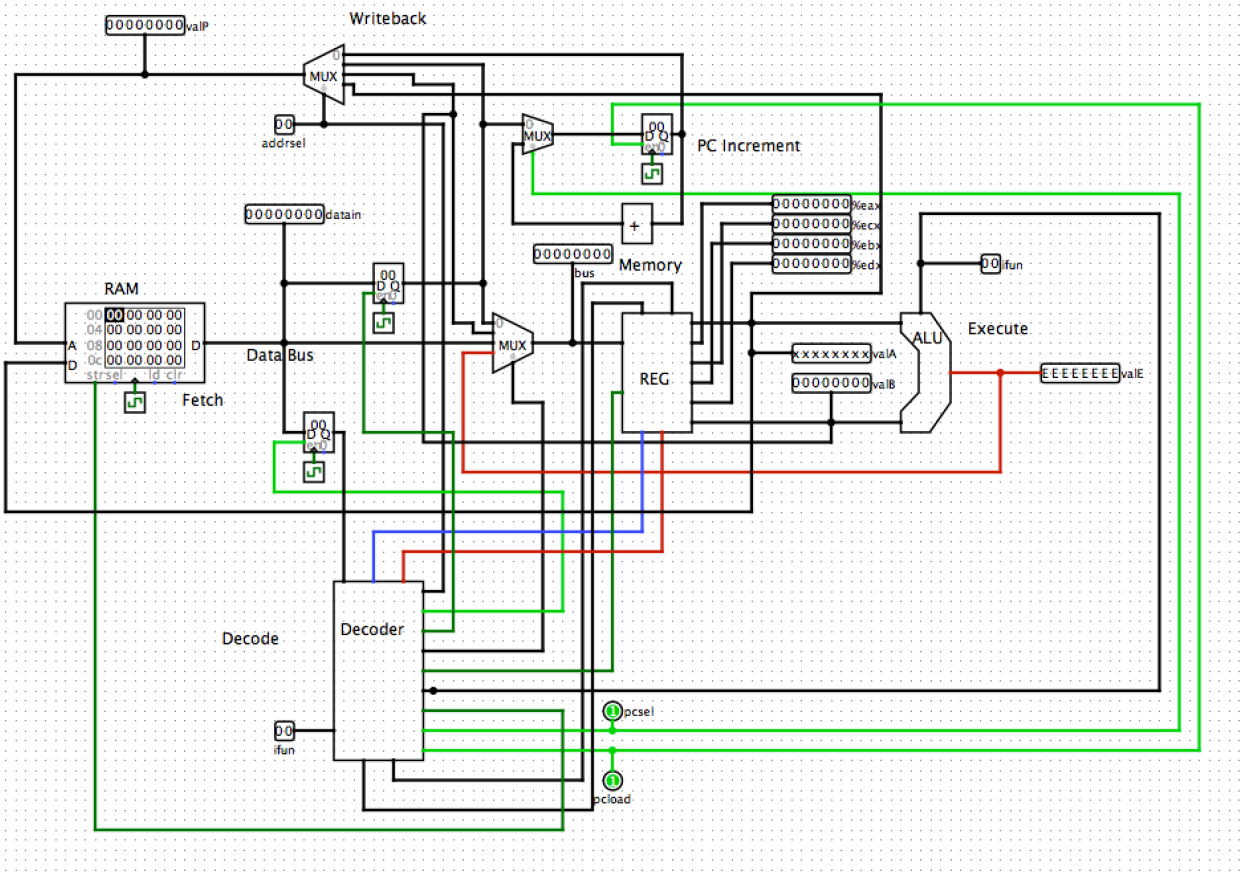
```
                        |
  0x0036: 30f005000000  |  irmovl $5, %eax
  0x003c: 30f300000000  |  irmovl $0, %ebx
  0x0042: 400310000000  |  rmmovl %eax, 16(%ebx)
  0x0048: 30f006000000  |  irmovl $6, %eax
  0x004e: 400314000000  |  rmmovl %eax, 20(%ebx)
  0x0054: 30f007000000  |  irmovl $7, %eax
  0x005a: 400318000000  |  rmmovl %eax, 24(%ebx)
  0x0060: 30f008000000  |  irmovl $8, %eax
  0x0066: 40031c000000  |  rmmovl %eax, 28(%ebx)
                        |
                        |
  0x006c: 500300000000  |  mrmovl 0(%ebx), %eax
  0x0072: 501310000000  |  mrmovl 16(%ebx), %ecx
  0x0078: 6001          |  addl %eax, %ecx
  0x007a: 401320000000  |  rmmovl %ecx, 32(%ebx)
                        |
  0x0080: 500304000000  |  mrmovl 4(%ebx), %eax
  0x0086: 501314000000  |  mrmovl 20(%ebx), %ecx
  0x008c: 6001          |  addl %eax, %ecx
  0x008e: 401324000000  |  rmmovl %ecx, 36(%ebx)
                        |
  0x0094: 500308000000  |  mrmovl 8(%ebx), %eax
  0x009a: 501318000000  |  mrmovl 24(%ebx), %ecx
  0x00a0: 6001          |  addl %eax, %ecx
  0x00a2: 401328000000  |  rmmovl %ecx, 40(%ebx)
                        |
  0x00a8: 50030c000000  |  mrmovl 12(%ebx), %eax      50 03 0c
  0x00ae: 50131c000000  |  mrmovl 28(%ebx), %ecx      50 13 1c
  0x00b4: 6001          |  addl %eax, %ecx            60 01
  0x00b6: 40132c000000  |  rmmovl %ecx, 44(%ebx)      40 13 2c
  0x00bc: 00            | halt
```

```
jaredwheeler — ssh -Y jaredwheeler94@compute.cse.tamu.edu — 80×24

[jaredwheeler94]@compute-linux1 ~/CSCE312/Project> (03:11:01 12/04/16)
:: ../sim/yis Project_Assembly.yo
Stopped in 35 steps at PC = 0xbc.  Status 'HLT', CC Z=0 S=0 O=0
Changes to registers:
%eax:   0x00000000      0x00000004
%ecx:   0x00000000      0x0000000c

Changes to memory:
0x0000: 0x0001f030      0x00000001
0x0004: 0xf3300000      0x00000002
0x0008: 0x00000000      0x00000003
0x000c: 0x00000340      0x00000004
0x0010: 0xf0300000      0x00000005
0x0014: 0x00000002      0x00000006
0x0018: 0x00040340      0x00000007
0x001c: 0xf0300000      0x00000008
0x0020: 0x00000003      0x00000006
0x0024: 0x00080340      0x00000008
0x0028: 0xf0300000      0x0000000a
0x002c: 0x00000004      0x0000000c

[jaredwheeler94]@compute-linux1 ~/CSCE312/Project> (03:11:12 12/04/16)
::
```

Matrix A: $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

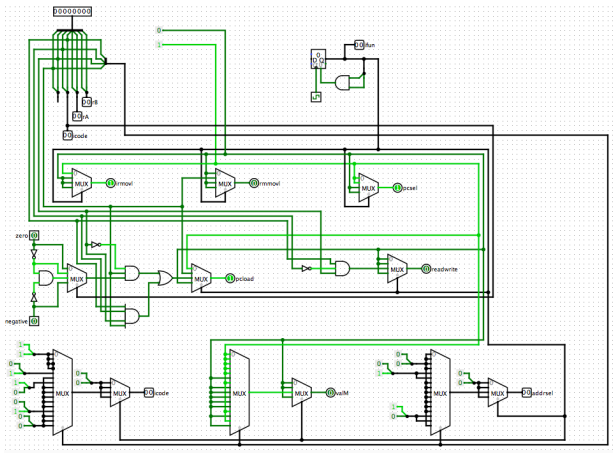
Matrix B: $\begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$

Matrix C: $\begin{pmatrix} 6 & 8 \\ 10 & 12 \end{pmatrix}$

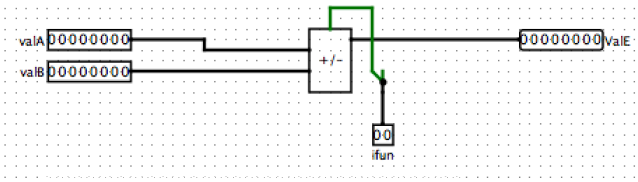


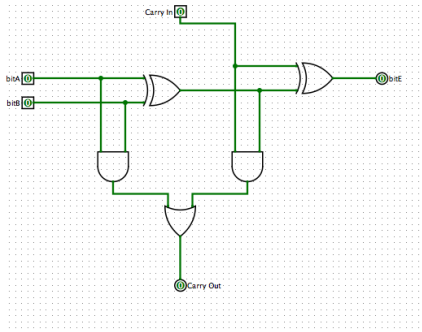
(No idea what is up with the red wire in this picture)
The Processor main: The design is based off of what was presented in the lecture powerpoint, featuring the clearly labeled Fetch, Decode, Execute, Memory, Writeback, and PC Increment sections.

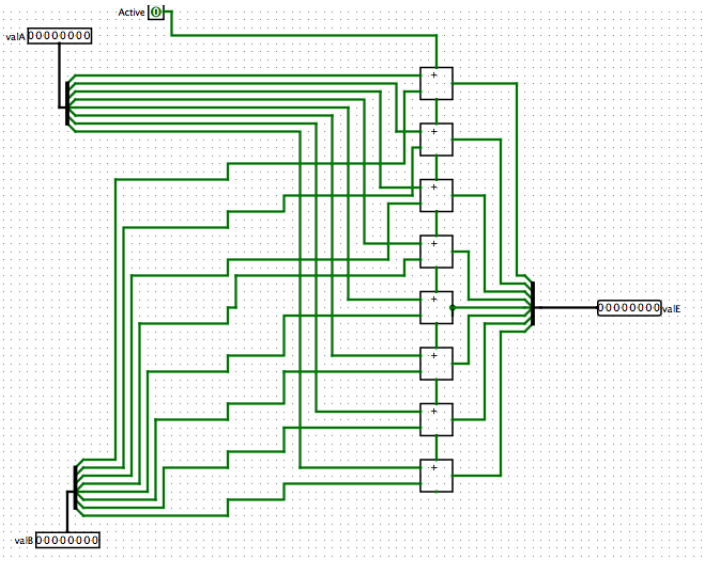
Fetch: Read the instruction memory from the assembly image that needs to be uploaded. It will send out the instructions that the image dictates.
Decode: Tells the program what to put into the registers, and what to do with it. In this program, the only things it can do is write to memory, load a register, and add registers. This is the sections that this will be decided.



Execute: The ALU consists of three circuits: the bit adder from lab 4, the eight bit adder from Lab 4, and a condensed ALU that is easier to look at. All of this particular circuit is pulled directly from Lab 4.







Memory: Gives a value in memory to write to.
Write Back: Writes the address in memory back to the registers.
PC Update: Increments the program counter by one to get the next instruction.