t 2016 Architecture - Rohan Pritchard’s Solutions

A lot of this stuff could well be wrong… Please leave comments if you find anything :)

## Question 1

### Part A

R = M[n]

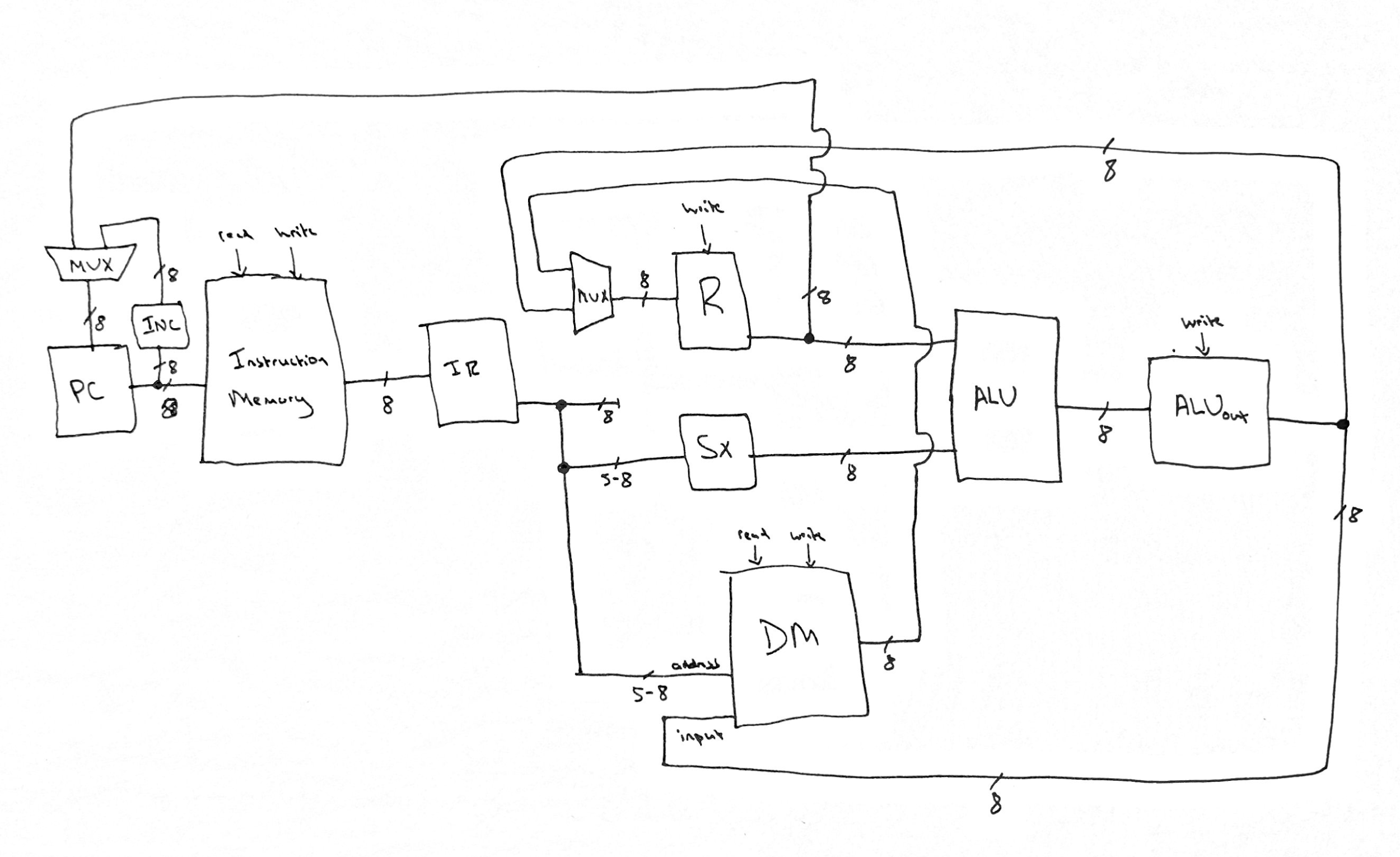
M[n] = R

R = R + n

PC = R

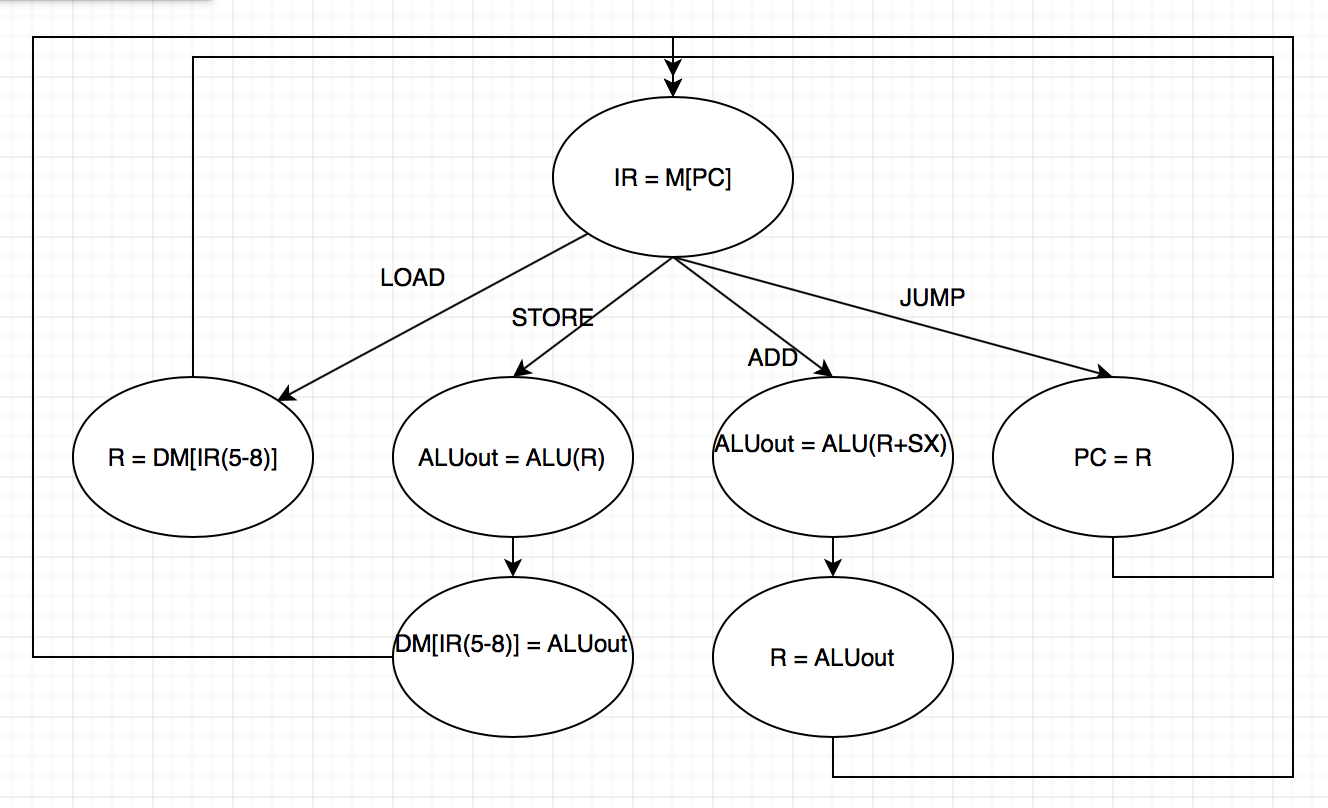
### Part B

5-8 bits should be 4-7.



### 

### Part C



^^Should be bits 4-7 not 5-8.

## Question 2

### Part A

You could store the local variables into reserved registers making them global variables, and the method would copy them from there (not recommended). The better way would be to push these variables to the stack (in the order: push c, push b, push a), and copy them from the stack in the method call.

### Part B

IEEE Single Precision = C1110000

Binary = -1001.0001

Decimal = -9.0625

### Part C

Decimal = -666.75

IEEE Single Precision Binary = 1 1000 1000 01001101011000000000000

IEEE Single Precision Hexadecimal = C426B000

### Let’s fail architecture yeet

### 

### 

### 

### 

### 

### 

### 

### Part D

rex:

push rbp ; setup pointer

mov rbp, rsp

push rbx ; for a

push rcx ; for b

push rdx ; for c

push r9 ; for d

push r10 ; for e

mov rbx, 3 ; a = 3

mov rcx, 2 ; b = 2

mov rdx, [rbp + 32] ; rdx = c (4th down on stack from base pointer: [rbp, rip, a, b, c]

mov r9, rcx ; d = b

sub r9, rdx ; d = b - c

Imul r9, 2 ; d = 2 \* (b -c)

rex\_while:

cmp r9, 0

je rex\_end\_while ; if d != 0, enter while loop

cmp r9, 0

jle rex\_else ; if d <= 0, go to else

add r9, -1 ; d = d - 1

jmp rex\_endif ; so we skip else statement

rex\_else:

add r9, 1 ; d = d + 1

rex\_endif:

push rdx ; c pushes to stack for next call

push rbx ; b pushes to stack for next call

push r9 ; d pushes to stack for next call

call rex

pop r9 ; take back off stack

pop rbx ; take back off stack

pop rdx ; take back off stack

mov r10, rax ; by caller convention, rax is return register, => e = rex(d, a, c)

jmp rex\_while

rex\_end\_while:

mov rax, r9 ; store return in rax, by caller convention. rax = d

pop r10 ; return to original state

pop r9

pop rdx

pop rcx

pop rbx

pop rbp

ret