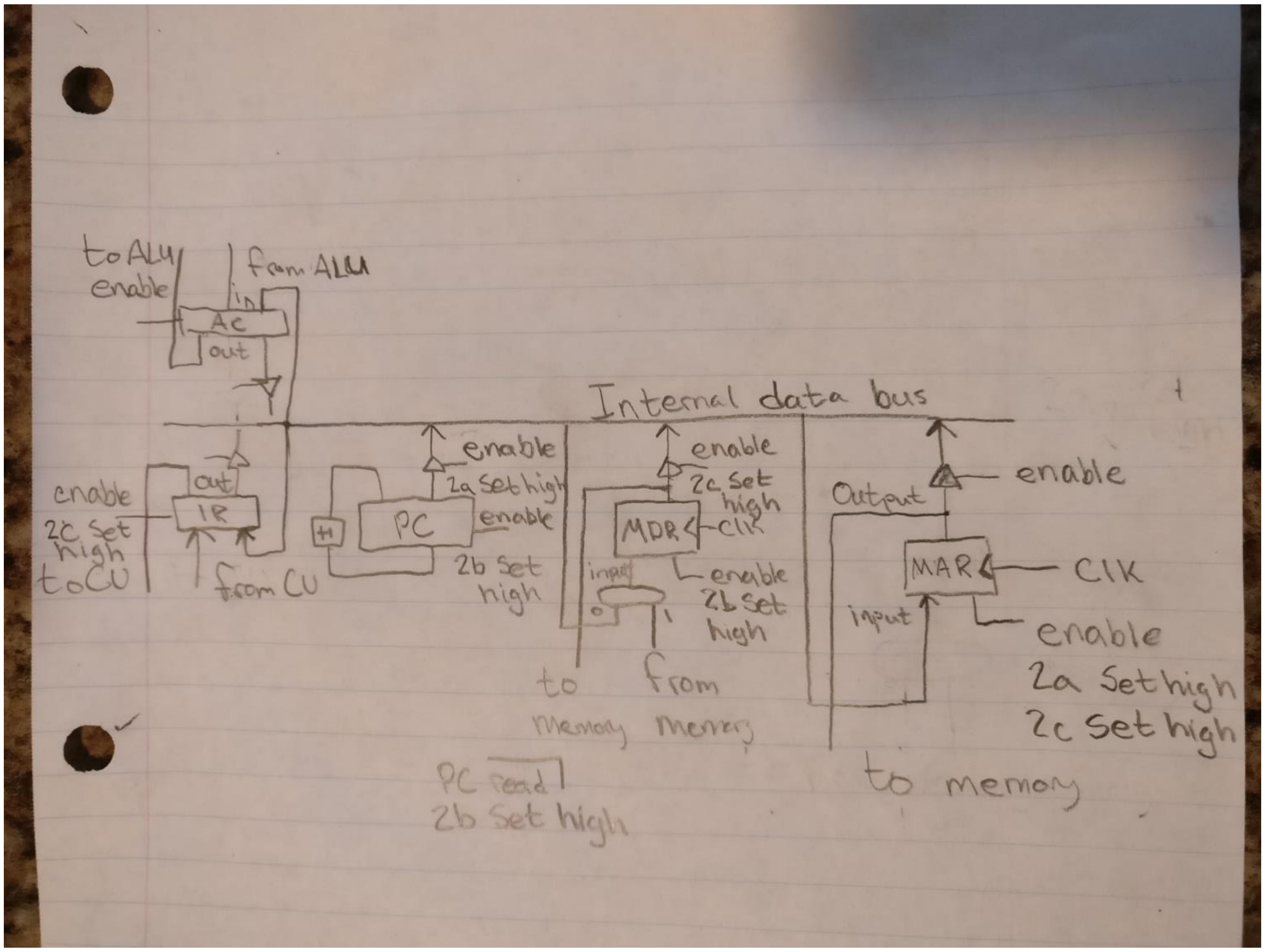
James Stallkamp

ECE 375 HW 1

1) CPU with 64 different instructions and memory of 1k(1024)
    a. determine the size of each memory works for the following instruction sets
        i. There are 64 different instructions hence the Op-code must be at least 6 bits long. 2^6 = 64. The memory is 1k so addressing must support at least 10 bits in order to address the full range of memory. 2^10 = 1024.
        ii. 3 Address Opcode(6) + 3 * Address(10) = 36 bits for a 3 address instructions
        iii. 2 Address Opcode(6) + 2 * Address(10) = 26 bits for a 2 address instructions
        iv. 1 Address Opcode(6) + 1 * Address(10) = 16 bits for a 1 address instructions
    b.

        Explain the advantages and disadvantages of three vs two address instruction formats.
        i. 3 Address: Three addresses allows for more powerful instructions, at the cost of high memory usage for each instruction.
        ii. 2 Address: Two addresses instructions take up less memory and therefore are more efficient. While still being complex enough for instructions like add, sub, and other math functions.
2) Consider Pseudo CPU instruction format is 16 bits which is divided into 4 bit opcode and 12 bit addressing fields and an instruction is stored in one memory word. Therefore MDR and AC are 16 bits and MAR and PC are 12 bits and IR is 4 bits
    a. Draw a diagram of the pseudo cpu clearly showing how MDR, AC, MAR, PC, and IR are interfaced with to the internal data bus using tri-state buffers and multiplexers including specific bit lines and control lines.
    b. For the diagram in part A indicate the control signals that need to be asserted(set to high) to implement each microperation for the fetch cycle shown below
        i. MAR <- PC
        ii. MDR <- M(MAR), PC <-PC+1
        iii. IR <- MDR(opcode), MAR<-MDR(address)



3) Based on the problem 2 indicate the control signals that need to be set to implement the microoperations for each of the following instructions. The immediate following lines are for the execute cycles. The fetch cycles are the same and are listed after.
   a. LDA x The following must be set to high 1) MAR enable and output, MDR enable 2) MDR output, AC enable
   b. STA x The following must be set to high. 1) MDR enable, PC read 2) MDR out, AC enable
   c. ADD x The following must be set to high. 1) MAR enable and output, MDR enable 2) MDR output, AC output and enable
   d. NAND x The following must be set to high. 1) MDR enable, AC out 2) MDR out AC out and enable
   e. J x The following must be set to high.) 1) MDR enable PC read. OR 2) MDR out AC enable and ADD for the ALU
4) Consider the 1 address instruction "addThenStoreIndirectWithPreDecrement" Instructions are 16 bits PC and MAR contain 12 bits AC MDR TEMP have 16 bits and IR is 4 bits. Original PC

should be preserved. Assume PC is already pointing to the addThenStore instruction and only PC and AC have the ability to increment themselves. Give the sequence of micro operations required to implement the execute cycle of the instruction. Fetch is given below.

a. Fetch
    i. MAR <- PC
    ii. MDR <-M(MAR), PC <- PC+1
    iii. IR <-MDR(opcode), MAR <-MDR(address)
b. Execute
    i. TEMP <- AC
    ii. MDR <- M[MAR] read effective address
    iii. PC <- MDR
    iv. PC <- PC - 1
    v. MDR <- PC
    vi. M[MAR] <- MDR
    vii. ?
    viii. ?
    ix. AC <- TEMP

5) Based on the initial circumstances shown below show how the contents are changed by the following instructions. Assume the instructions are run independently. IE the contents are reset to the initial state before running the next instruction. For each instruction a smaller table is given showing which cells are changed in the initial tables. Arrow indicates change.

a. Sts $0107, r29 writes the contents of register 29 to data space location $0107. Register 29 is YH and in this case is 01.

address	Data memory
X	02->01

b. Ld r3, -x loads. First a pre decrement is performed on x then register 3 is loaded with the where register x is now pointing to. X was pointing at data space 0106, that becomes 0105. Register 3 is loaded with 2D

registers	contents
R3	07->2D

c. Rol r2 performs a logical rol on the contents of r2. 1b in binary is 00011011, logical shift left with a c flag(because sreg is FF) becomes 00110111 which is 37 in hex.

registers	contents
R2	1b -> 37

d. Mul r1, r3 multiples the contents of register 1 and 3, then storing the result in register 1. 5*7 = 35, 23 in hex

registers	contents
R1	05->23

e. Sbr r26, 7. Set bits in register. This performs the logical ORI between the contents of the register and the constant. R26 is XL or 06. 00000110 || 00000111 -> 00000111. So register 26 becomes 7

registers	contents
x	0107

	Registers
R0	01
R1	05
R2	1B
R3	07
R4	01
X	0106
Y	0102
SREG	FF

	Data Memory
0100	01
0101	BE
0102	35
0103	EC
0104	48
0105	2D
0106	04
0107	02