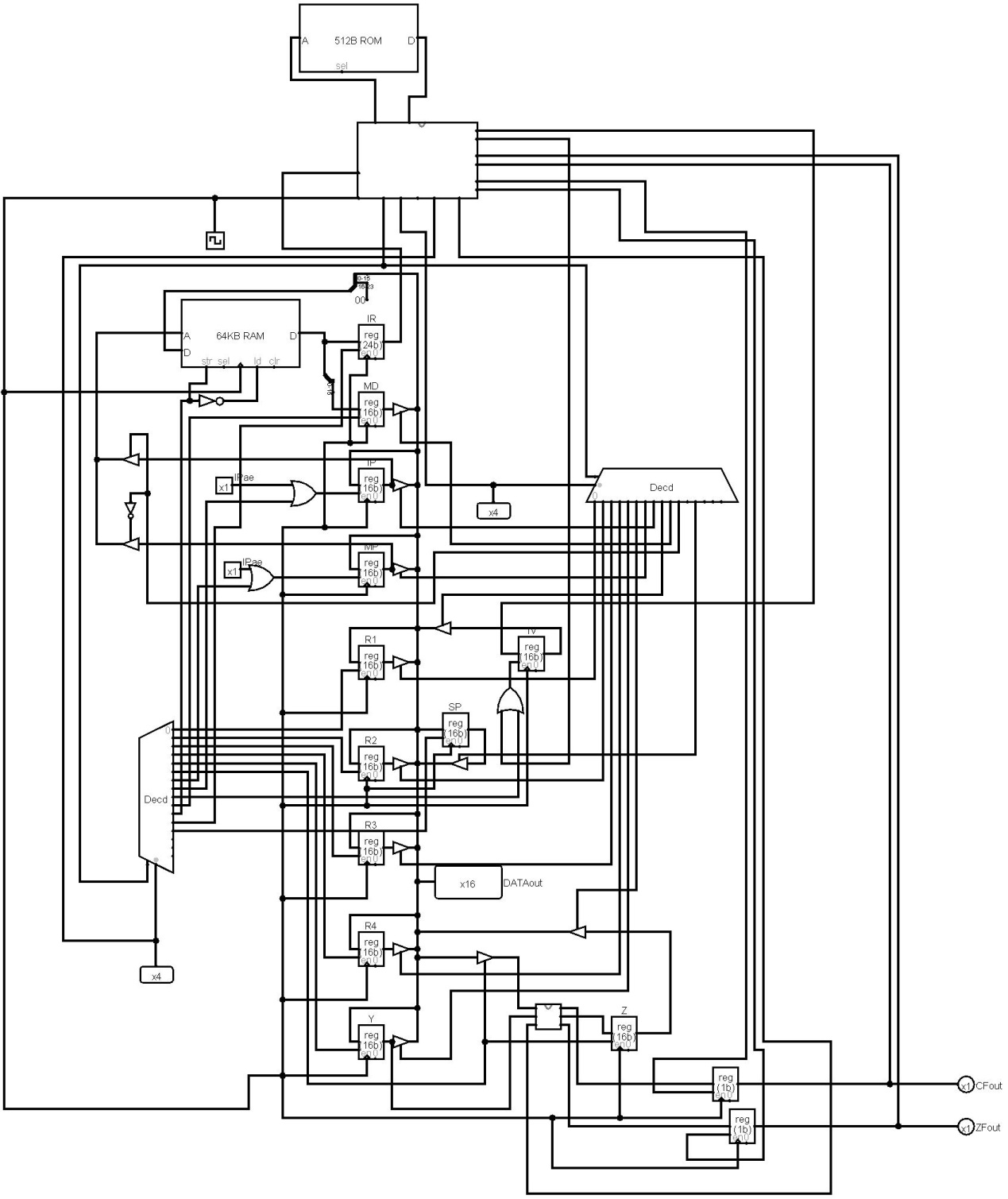
Single Bus Processor Design

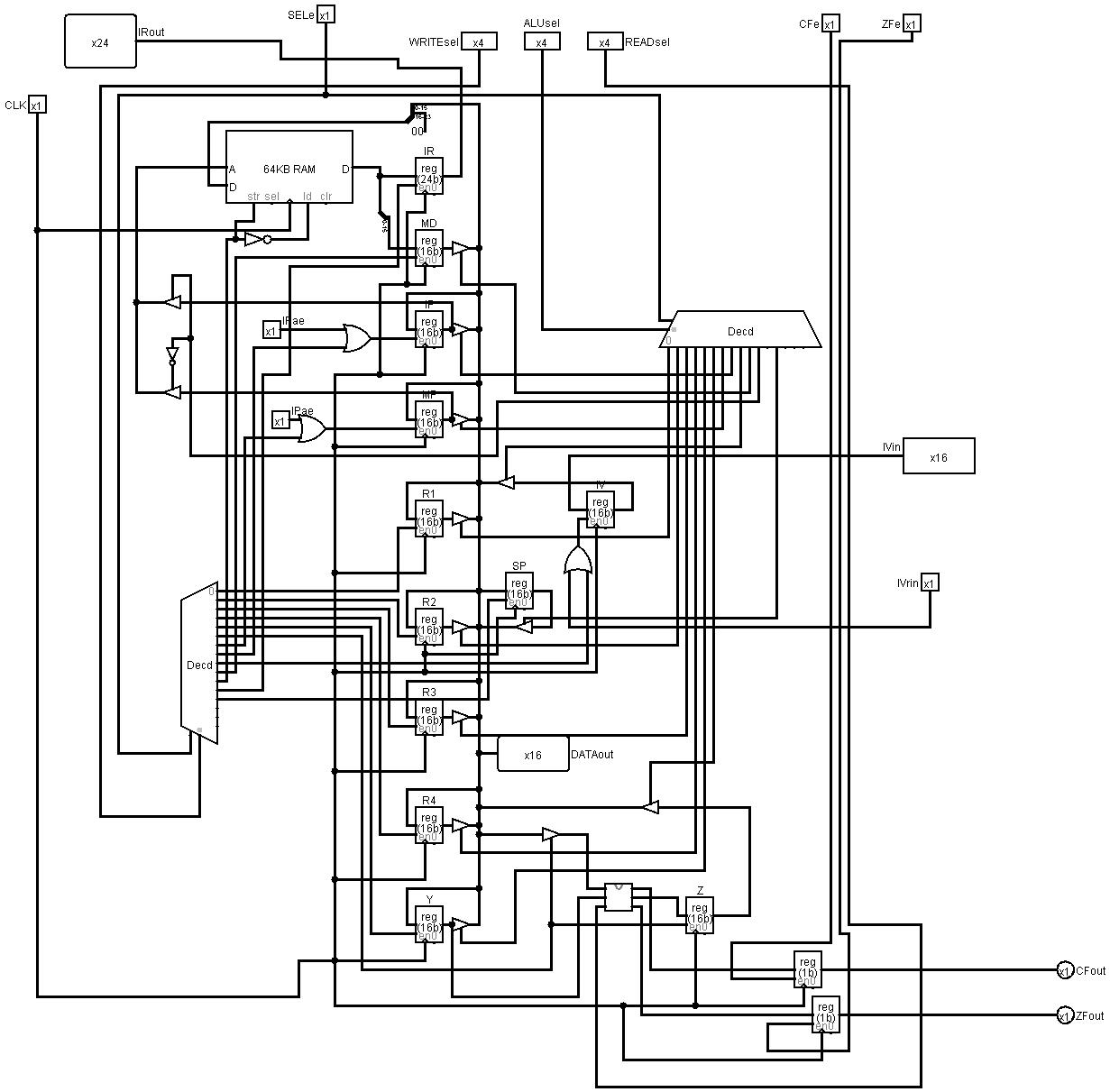
COE205 LAB, Project 1

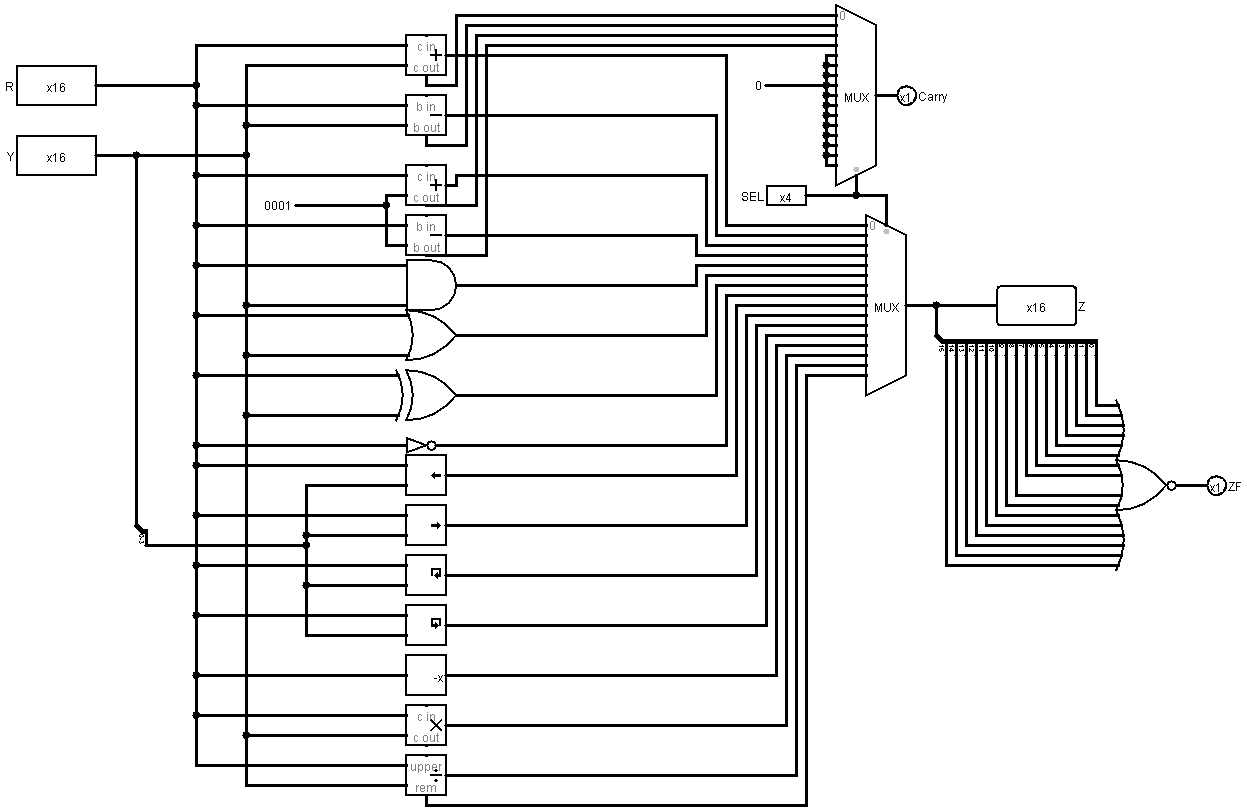
By: Mahmoud Elshafei Ahmed

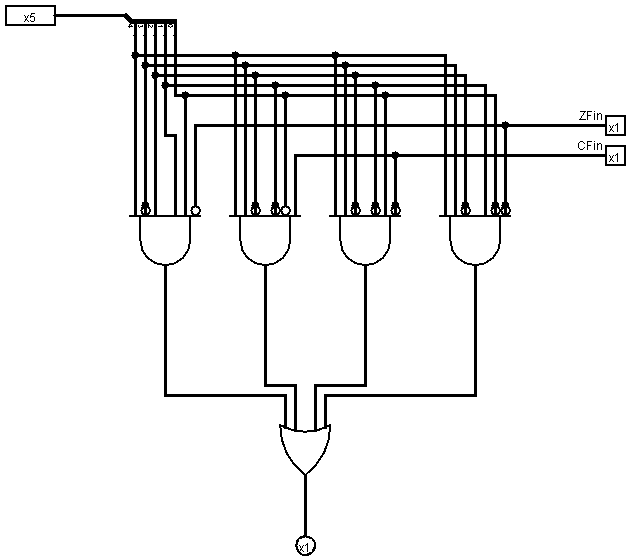
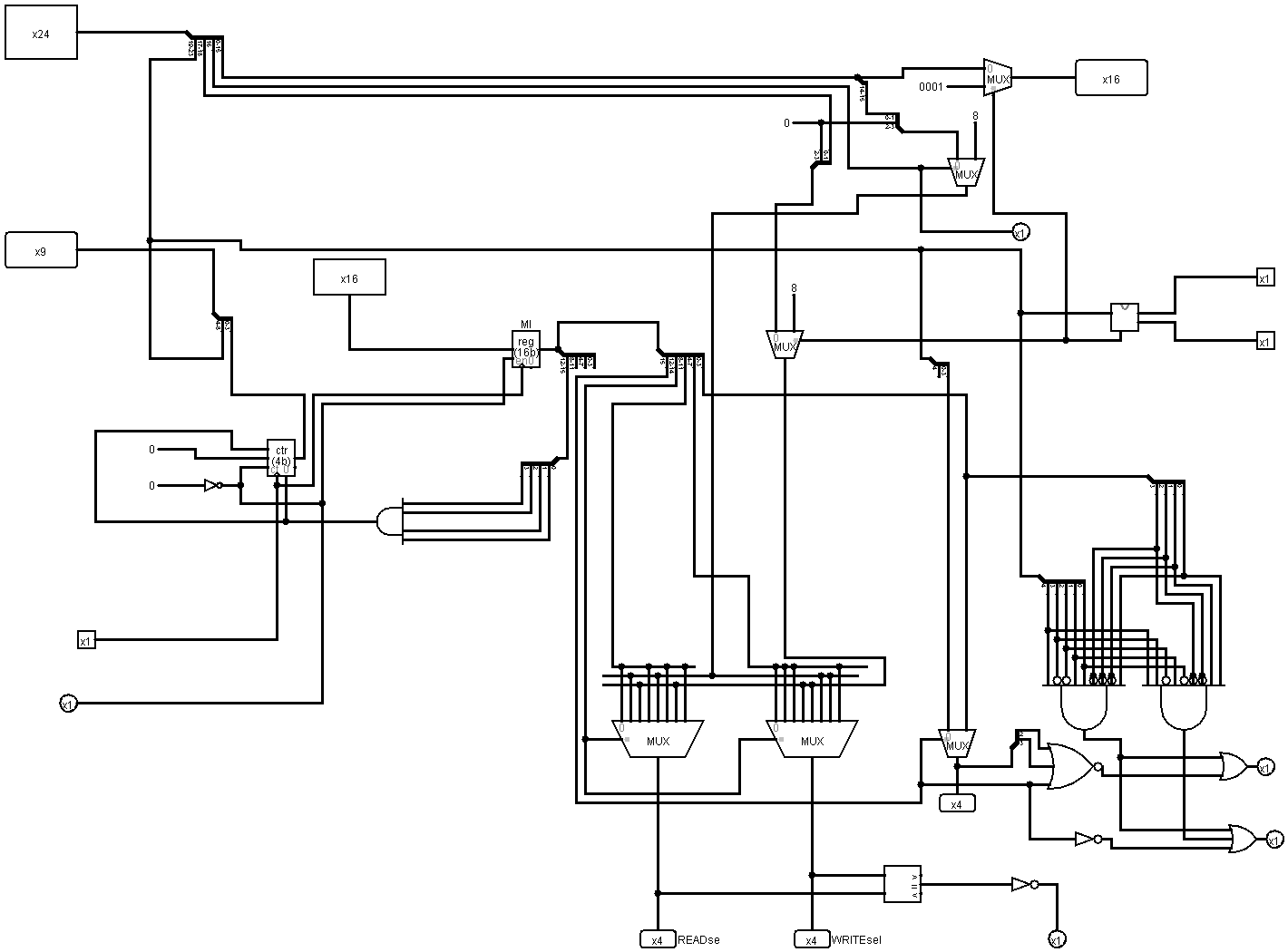


Introduction:

The aim of this project was to provide a practical application to some of the concepts discussed in lectures and labs. The project is to design and implement a basic single bus CPU. This would allow the student to better understand the theory of this course.







Understanding the report:

|  |  |  |  |
| --- | --- | --- | --- |
| Written | Meaning | Written | Meaning |
| Ds | Destination register selected in the machine code | IP, SP, mem | Instruction pointer, stack pointer, memory |
| Sc | Source register selected in the machine code, can also be immediate value | ALU, ALUsel | Arithmetic logic unit, alu selection bit |
| Reg | Register selected by the microinstruction code | ZF, ZFe, CF, CFe | Zero flag, ZF enable bit, carry flag, CF enable bit |

Micro instruction select component:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Micro instruction ROM  Select bits | | READ | WRITE | ALUins | CFe | ZFe |
| 0 | 0000 | REG | REG | 1 | 1 | 1 |
| 1 | 0001 | SC | REG | 1 | 1 | 1 |
| 2 | 0010 | DS | REG | 1 | 1 | 1 |
| 3 | 0011 | REG | DS | 1 | 1 | 1 |
| 4 | 0100 | SC | DS | 1 | 1 | 1 |
| 5 | 0101 | reg | sc | 1 | 1 | 1 |
| 6 | 0110 | ds | sc | 1 | 1 | 1 |
| 7 | 0111 | REG | REG | 1 | 1 | 1 |
| 8 | 1000 | REG | REG | 0 | 0 | 0 |
| 9 | 1001 | SC | REG | 0 | 0 | 0 |
| A | 1010 | DS | REG | 0 | 0 | 0 |
| B | 1011 | REG | DS | 0 | 0 | 0 |
| C | 1100 | SC | DS | 0 | 0 | 0 |
| D | 1101 | reg | sc | 0 | 0 | 0 |
| E | 1110 | Ds | sc | 0 | 0 | 0 |
| F | 1111 | REG | REG | 0 | 0 | 0 |

Micro instruction format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bit 15 | 14-12 | 11-8 | 7-4 | 3-0 |
| alu sel location bit | Reg location sel | Read sel | Write sel | Alu sel |
| 1 indicates alu control will be from the machine code | Selects the read and write locations according to the table above | If read indicates register then will read from the register specified here | If write indicates register then will write to the register specified here | If alu select location is 1 then will read alu from here |
| Example: | | | | |
| 8 | | 7 | 5 | 2 |
| Select alu from micro instruction code | Select both input and output from micro instruction code | Read from IP | Write to Z (arithmetic operation) | Alu sel will be 2 (increment) |

Example instructions:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| instruction | Machine code | Binary operation sel | Destination sel | Immediate sel | Source/immediate |
| ADD R3, R1 | 040000 | 00000 | 10 | 0 | 00b |
| SUB R1, R2 | 084000 | 00001 | 00 | 0 | 01b |
| XCH R2, R1 | 884000 | 10001 | 00 | 0 | 01b |
| ADD R3, R1 | 040000 | 00000 | 10 | 0 | 00b |
| ADD R2, 500h | 030500 | 00000 | 01 | 1 | 0500h |
| JMP -5 | 91fffb | 10010 | XX | 1 | FFFBh |

Instruction set:

|  |  |  |
| --- | --- | --- |
| Instruction code |  | notes |
| 00000 | Add | Ds 🡨 ds + sc, zf and cf affected |
| 00001 | Sub | Ds 🡨 ds - sc, zf and cf affected |
| 00010 | Inc | Ds 🡨 ds + 1, zf and cf affected |
| 00011 | Dec | Ds 🡨 ds - 1, zf and cf affected |
| 00100 | And | Ds 🡨 ds AND sc, zf affected |
| 00101 | Or | Ds 🡨 ds OR sc, zf affected |
| 00110 | Xor | Ds 🡨 ds XOR sc, zf affected |
| 00111 | Not | Ds 🡨 ds NOT sc, zf affected |
| 01000 | Shl | Ds 🡨 ds SHL sc, zf affected |
| 01001 | Shr | Ds 🡨 ds SHR sc, zf affected |
| 01010 | Rol | Ds 🡨 ds ROL sc, zf affected |
| 01011 | Ror | Ds 🡨 ds ROR sc, zf affected |
| 01100 | Neg | Ds 🡨 - ds, zf affected |
| 01101 | Mul | Ds 🡨 ds \* sc, zf affected |
| 01110 | Div | Ds 🡨 ds / sc, zf affected |
| 01111 | Mod | Ds 🡨 ds % sc, zf affected |
| 10000 | Mov | Ds 🡨 sc |
| 10001 | xchg | Ds 🡨 sc, sc 🡨 ds |
| 10010 | Jmp | IP 🡨 IP + sc |
| 10011 | cmp | ds – sc, zf and cf affected |
| 10100 | Mov(mem to ds) | Ds 🡨 mem, sc used as mem address |
| 10101 | Mov(ds to mem) | Mem 🡨 ds, sc used as mem address |
| 10110 | Xchg(mem & ds) | Ds 🡨 mem, Mem 🡨 ds, sc used as mem address |
| 10111 | Jz | IP 🡨 IP + sc, if zf=1 else: IP 🡨 IP + 1 |
| 11000 | Ja | IP 🡨 IP + sc, if cf=0 else: IP 🡨 IP + 1 |
| 11001 | Jb | IP 🡨 IP + sc, if cf=1 else: IP 🡨 IP + 1 |
| 11010 | Loop | R3 🡨 r3 – 1, IP 🡨 IP + sc if zf=1 else: IP 🡨 IP + 1 |
| 11011 | Push | Mem 🡨 ds, SP 🡨 SP – 1, SP used as mem address |
| 11100 | Pop | Ds 🡨 mem, SP 🡨 SP + 1, SP used as mem address |
| 11101 | Call | Mem 🡨 IP, IP 🡨 sc, SP 🡨 SP – 1, SP used as mem address |
| 11110 | Ret | SP 🡨 SP + 1, IP 🡨 mem, SP used as mem address, IP 🡨 IP+1 |
| 11111 | Writehex | Incomplete |

Fibonacci series:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| adress | Instruction | Binary operation sel | Destination sel | Immediate sel | Source/ immediate | Machine code |
| 8000 | Mov r3, 0 | 10000 | 10 | 1 | 0000 | 850000 |
| 8001 | call Fib | 11101 | xx | 1 | 8004 | E98005 |
| 8002 | inc r3 | 00010 | 10 | 0 | 0000 | 140000 |
| 8003 | Mov [r3], r1 | 10101 | 00 | 0 | 1000 | A88000 |
| 8004 | jmp -3 | 10010 | 00 | 1 | fffe | 91fffd |
| 8005 | Fib: cmp r3, 0 | 10011 | 10 | 1 | 0000 | 9d0000 |
| 8006 | jz iff0 | 10111 | xx | 1 | 000e | B9000e |
| 8007 | cmp r3, 1 | 10011 | 10 | 1 | 0001 | 9d0001 |
| 8008 | jz iff1 | 10111 | Xx | 1 | 000e | B9000e |
| 8009 | push r3 | 11011 | 10 | 0 | 0000 | Dc0000 |
| 800a | push r2 | 11011 | 01 | 0 | 0000 | Da0000 |
| 800b | dec r3 | 00011 | 10 | 0 | 0000 | 1c0000 |
| 800c | call Fib | 11101 | Xx | 1 | 8004 | E98005 |
| 800d | mov r2, r1 | 10000 | 01 | 0 | 0000 | 820000 |
| 800e | dec r3 | 00011 | 10 | 0 | 0000 | 1c0000 |
| 800f | call Fib | 11101 | Xx | 1 | 8004 | E98005 |
| 8010 | add r1, r2 | 00000 | 00 | 0 | 4000 | 004000 |
| 8011 | pop r2 | 11100 | 01 | 0 | 0000 | E20000 |
| 8012 | pop r3 | 11100 | 10 | 0 | 0000 | E40000 |
| 8013 | Ret | 11110 | Xx | 0 | 0000 | F00000 |
| 8014 | Iff0: mov r1, 0 | 10000 | 00 | 1 | 0000 | 810000 |
| 8015 | Ret | 11110 | Xx | 0 | 0000 | F00000 |
| 8016 | Iff1: mov r1, 1 | 10000 | 00 | 1 | 0001 | 810001 |
| 8017 | ret | 11110 | xx | 0 | 0000 | F00000 |
| 8018 |  |  |  |  |  |  |

mov r3, 0

call Fib

inc r3

mov [r3], r1

jmp -2

fib:

cmp r3, 0

jz iff0

cmp r3, 1

jz iff1

push r3

push r2

dec r3

call Fib

mov r2, r1

dec r3

call Fib

add r1, r2

pop r2

pop r3

ret

iff0:

mov r1, 0

ret

iff1:

mov r1, 1

ret

Instructions can be copied directly to logisim ram module starting at address ‘8000h’:

850000 E98005 140000 A80000 91fffd 9d0000 B9000e 9d0001 B9000e Dc0000 Da0000 1c0000 E98005 820000 1c0000 E98005 004000 E20000 E40000 F00000 810000 F00000 810001 F00000

It is require to set the IP (instruction pointer) to ‘8000h’ and SP to ‘ffffh’

This program will write to memory the Fibonacci series starting from address ‘0000h’

Conclusion:

I was able to design and implement a simple CPU using the knowledge from the lectures and labs. The CPU designed currently has 31 working instructions including CALL and RET. Due to simplicity of Logisim making big changes to my design can be easily done without a need for overhauls. In designing, implementing and ultimately running this CPU I was able to better understand the workings of certain concepts like stacks and jumps. In the future I wish to better understand other, more advanced concepts such as pipelining and multi-bus systems.