COMP 303 TERM PROJECT

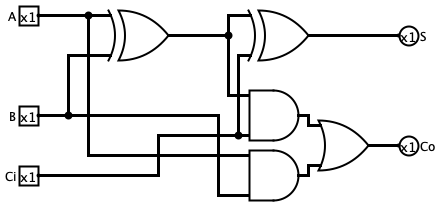
FALL 2019

Single Cycle Processor Design

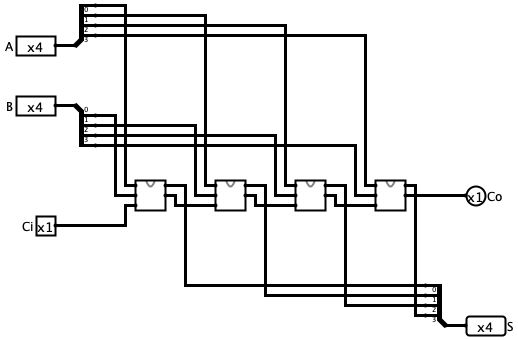
Murat Erdoğan – Tolga Sümer

**ALU**

Our instructions are 32-bit, using the same instruction format as MIPS. All the components except the adder is used from built in components.To make a 16-bit adder we first had to implement a 1- bit full adder.



Then we have to make a 4-bit with 4 1-bit full adder.

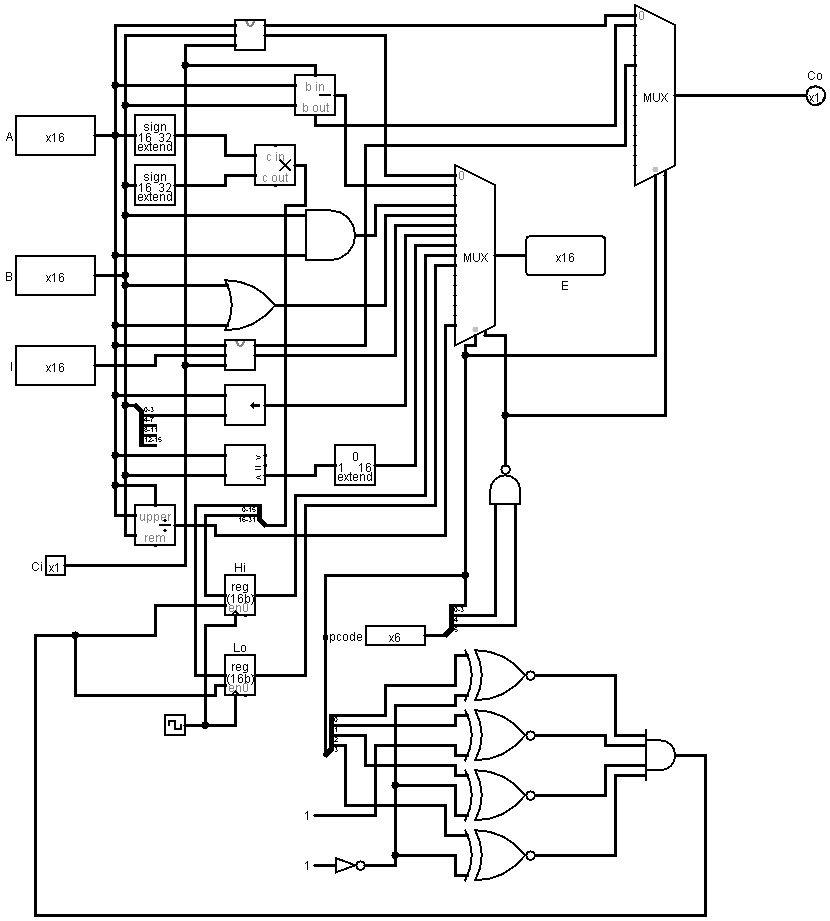


At last, with 4 4-bit adders we can implement a 16-bit full adder.

A screenshot of a cell phone

Description automatically generated

After combining the other parts, we complete our ALU.



**Register File**

We designed a register file with sixteen 16-bit registers. The register file supports the same I/O signals as the MIPS register file with two read register address ports, two read register output ports, one write register address port, one write register data port, a register write control signal and a clock input.

A circuit board

Description automatically generated

**Data Memory**

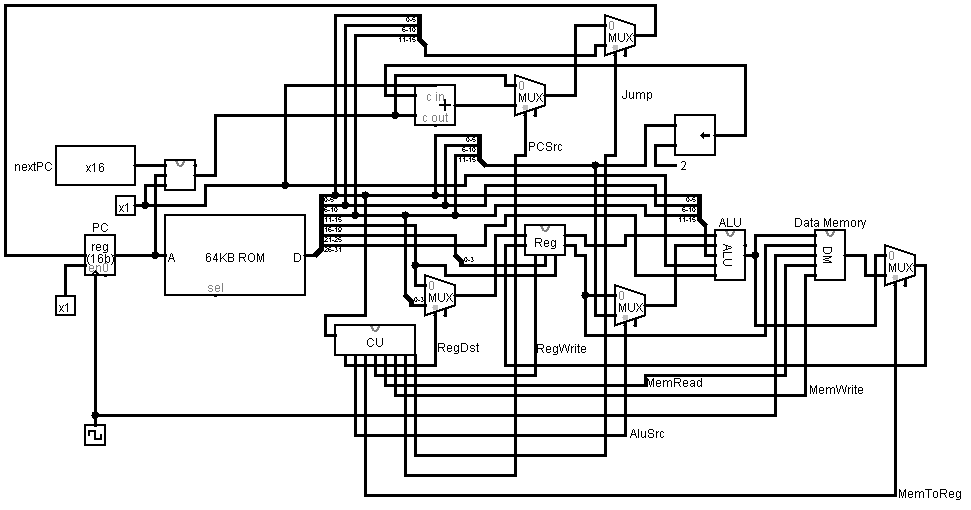
We built a data memory by using the RAM module.

A screenshot of a cell phone

Description automatically generated

**CPU**

By combining our CPU, ALU and data memory we completed our CPU.



We also made a control unit.

A drawing of a hanger

Description automatically generated

**Control Units table:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **regDst** | **aluSrc** | **memToReg** | **regWrite** | **memRead** | **memWrite** | **PCSrc** | **Jump** | **opcode** |
| add | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000000 |
| sub | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000001 |
| mult | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 000010 |
| and | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000011 |
| or | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000100 |
| addi | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 000101 |
| sll | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000110 |
| slt | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 000111 |
| mfhi | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 001000 |
| mflo | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 001001 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 001010 |
| sw | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 001011 |
| beq | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 001100 |
| blez | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 001101 |
| j | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 001110 |
| myIns | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 001111 |

**Note:** In our design, we had two registers in our ALU which stored the result of mult operation, when we call our mfhi and mflo operations the result of the multiplication is called.

|  |  |
| --- | --- |
| Assembly Code | Address of Instruction |
| addi $0, $0, 0 | 00010100000000000000000000000000 |
| addi $1, $1, 0 | 00010100001000010000000000000000 |
| addi $2, $2, 0 | 00010100010000100000000000000000 |
| addi $9, $9, 4 | 00010101001010010000000000000100 |
| addi $7, $7, 2 | 00010100111001110000000000000010 |
| beq $9, $0, 17 | 00110001001000000000000000010001 |
| lw $3, 0($0) | 00101000000000110000000000000000 |
| lw $4, 0($1) | 00101000001001000000000000000000 |
| add $5, $4, $3 | 00000000101001000001100000000000 |
| mult $5, $5, $7 | 00001000101001010011100000000000 |
| mflo $6 | 00100100000000000010100000000000 |
| sw $6, 0($2) | 00101100010001010000000000000000 |
| addi $0, $0, 1 | 00010100000000000000000000000001 |
| addi $1, $1, 1 | 00010100001000010000000000000001 |
| addi $2, $2, 1 | 00010100010000100000000000000001 |
| j 6 | 00111000000000000000000000000101 |

Our additional instruction is the division operation. We used the built-in function.