

**CSE 2114 Project Report**

**Project Report on**

**Course No:**2114

**Course Title**: Computer Architecture Laboratory

**Topic**: Design of a minimal computer system of 22 bit

Submitted by:

Adit Mugdha Das

Roll:2107118

**Under the Guidance of:**

**Dr. M.M.A. Hashem**

**Professor, Department of Computer Science and Engineering,**

**KUET**

**Md. Badiuzzaman Shuvo**

**Lecturer, Department of Computer Science and Engineering,**

**KUET**

**Khulna University of Engineering and Technology**

**Objectives:**

1. To know about the design of CPU.

2. To implement a simple CPU in Logisim software and simulate the

Results.

**Introduction :**

The Central Processing Unit (CPU) is the primary component of a computer that acts as its “control center.” The CPU, also referred to as the “central” or “main” processor, is a complex set of electronic circuitry that runs the machine's operating system and apps . I have made a 22 bit Cpu using 22 bit Alu, which can perform Addition,Subtraction,Bitwise OR,Bitwise AND,Bitwise NAND,Bitwise NOR. The cpu has also program for Store,Jump,Load,Halt. The control unit handles all the operation using rom which stores the required instruction code.

As I have added two universal gate operation I have to add more instruction in rom. So my rom code is slightly different from our teachers circuit. Here is the modified rom code :

09 19 14 29 19 31 02 29

19 39 02 29 51 59 02 29

19 41 02 29 19 49 02 21

02 02 02 29 19 71 02 29

19 61 02 29 19 69 02 78

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

00 00 00 00 00 00 00 00

Here I have used my last 4 bit as opcode(21,20,19,18) and other remaining bits as data bits. As my logisim ram shows 24 bits to input I have used 22th and 23th bit as don’t care(0,0) . So the written opcode for every operation get changed in the case of giving input because I have to use 8 bit to give my opcode as perfect input in the ram.

In the next page the opcode for every operation and its equivalent input opcode is given:

Here are the opcodes for every operation

|  |  |
| --- | --- |
| **NAME OF THE OPERATION** | **OPCODE** |
| BITWISE AND | 0 |
| ADD | 1 |
| STORE | 2 |
| BITWISE OR | 3 |
| SUBTRACT | 4 |
| JUMP | 5 |
| LOAD | 6 |
| NAND | 7 |
| NOR | 8 |
| HALT | 9 |

**Now the equivalent machine opcode for every operation is given**

**FOR ADDITON:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** | **CALCULATTION** |
| **000000** | **Load** | **18000f** | **f=20** |
| **000001** | **Add(f+e)** | **04000c** | **e=10** |
| **000002** | **Store(a)** | **08000a** | **a=30** |
| **000003** | **Halt** | **240000** |  |

**FOR SUBTRACTION:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** | **CALCULATTION** |
| **000000** | **Load** | **18000f** | **f=50** |
| **000001** | **Sub(f-e)** | **10000e** | **e=20** |
| **000002** | **Store(a)** | **08000a** | **a=30** |
| **000003** | **Halt** | **240000** |  |

**FOR AND:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** | **CALCULATTION** |
| **000000** | **Load** | **18000f** | **f=4** |
| **000001** | **And(f,e)** | **00000e** | **e=4** |
| **000002** | **Store(a)** | **08000a** | **a=4** |
| **000003** | **Halt** | **240000** |  |

**FOR OR:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** | **CALCULATTION** |
| **000000** | **Load** | **18000f** | **f=4** |
| **000001** | **Or(f,e)** | **00000e** | **e=2** |
| **000002** | **Store(a)** | **08000a** | **a=6** |
| **000003** | **Halt** | **240000** |  |

**FOR NOR:**

|  |  |  |
| --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** |
| **000000** | **Load** | **18000f** |
| **000001** | **Nor(f,e)** | **00000e** |
| **000002** | **Store(a)** | **08000a** |
| **000003** | **Halt** | **240000** |

**FOR NAND:**

|  |  |  |
| --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** |
| **000000** | **Load** | **18000f** |
| **000001** | **Nand(f,e)** | **1c000e** |
| **000002** | **Store(a)** | **08000a** |
| **000003** | **Halt** | **240000** |

**FOR JUMP:**

|  |  |  |
| --- | --- | --- |
| **ADDRESS** | **ASSEMBLY CODE** | **MACHINE CODE** |
| **000000** | **Load** | **18000f** |
| **000001** | **Add(f+e)** | **1c000e** |
| **000002** | **Store** | **08000a** |
| **000003** | **Jump** | **14001f** |
| **00001f** | **Halt** |  |

**Discussion**:

While developing the ALU, the use of multiplexer determines which logic circuit is to be used. Using some logics, the zero value of the register was also determined Carry, overflow and negative result was also determined. This project involves designing an original instruction set, devising a data path and control unit, and writing a simulator for a processor. There are in fact several ways to further improve the design. The first method is using caches. This greatly reduces the time taken on memory access. Improving efficiency on memory access will not improve the overall performance. However, memory access takes much longer than ALU and bottleneck will be on it and caches will be very helpful to improve the performance.