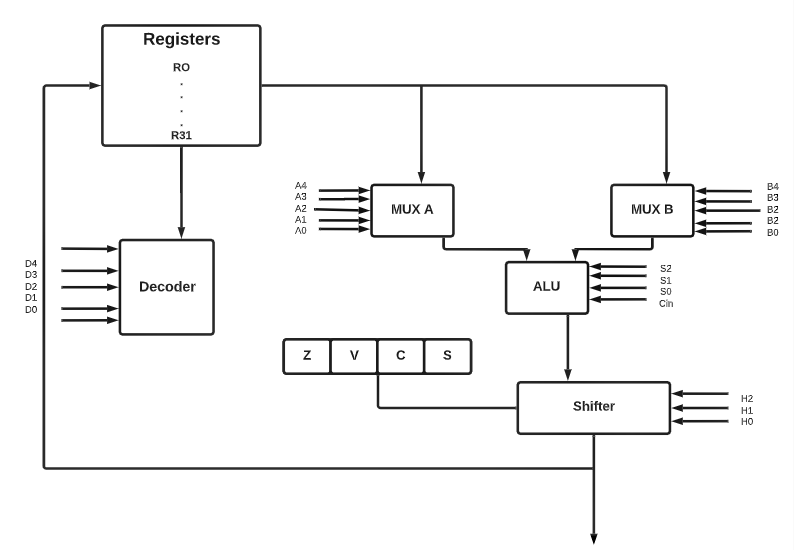
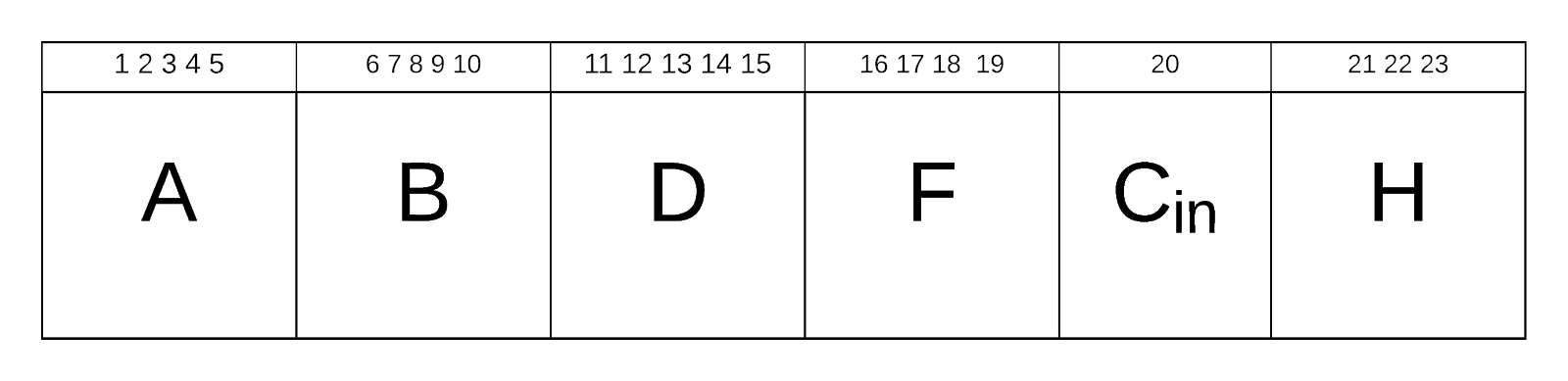
**ICS3C-111841-Edwin Irungu**

**GROUP 9**

**i)Draw the diagram representing the processor logic (use MS Visio).**

****

**ii)**

**iii)Write two programs to perform given operations of your own choice. Your programs should be in machine, register transfer, assembly, high level and natural languages besides having at least 8/16 instruction lines each representing 8/16 micro-operations.**

**High Level Language. Language used is java.**

import java.util.Scanner;

class Subtract

{

   public static void main(String args[])

   {

      int p, q, r;

      System.out.println("Enter two integers to calculate their subtraction: ");

      Scanner sc = new Scanner(System.in);

      p = sc.nextInt();

      q = sc.nextInt();

      r = p - q;

      System.out.println("subtraction of entered integers = "+r);

   }

}

**Natural Language**

function subtract(){

p= 14;

q= 2;

r = p-q;

return r;

}

**Assembly language**

R14<--R6-R4

        ;STORE THE VALUES

MVI B, 06H

MVI C, 04H

;MOVE CONTENTS INTO ACCUMULATOR

MOV A,B

SUB C    ;ADD THE CONTENTS

STA 0EH

HLT

**Register Transfer language**

R6 <- R3

R4<- R7

R14<--R6-R4

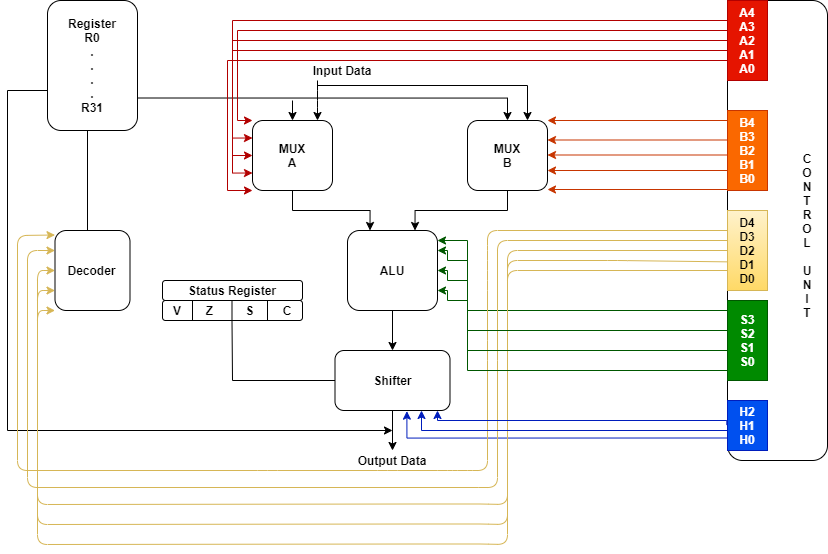
**Machine Language**

0000000000000000

**iv)  Represent your programs using a table to include ROM locations and next addresses**

|  |  |  |
| --- | --- | --- |
|  | ROM | PC |
| R6 <- R3 | 00 | 01 |
| R4<- R7 | 01 | 10 |
| R14<--R6-R4 | 10 | (00 if you’re repeating the steps and we use HALT to stop execution) |

**v) Draw the processor logic together with the control unit clearly indicating the control lines as connected to the processor unit.**

****

vi) **Based on the above questions, explain the relation between the processor unit, the control unit, machine language, register transfer language, assembly language, high-level language and natural language. You can use an appropriate example to explain the relation.**

A natural language is a human language such as English for instance. For any particular user before writing code to execute any instruction, they formulate a reasoning based on a language they understand. For example, a user would like to perform an arithmetic operation. To do this the first statement that is created is “Five plus Four”. This statement is in a natural language format.

Pseudocode:

Start

Initialize A to zero

Initialize B to zero

Initialize Sum to zero

Output “Enter Numbers”

Input first number to A

Input second number to B

Sum = A + B

Output “Sum = ” value of Sum

Stop

Next for this instruction to be introduced into a computer, it is converted to a high level language by the user through various programming languages such as PHP. An example of such a high level language (PHP) to perform addition of 2 numbers is as shown:

**<form action=”PHP\_SELF()” method=”POST”>**

**First Number: <input type=”number” name=”fnumber”>   <br>**

**Second Number: <input type=”number” name=”snumber”>   <br>**

**<input type=”submit” name=”submit” value=”Calculate”>   <br>**

**</form>**

**<?php**

**$A=$\_POST(‘fnumber’);**

**$B=$\_POST(‘snumber’);**

**$Sum=$A+$B;**

**echo "Sum: ",$Sum;**

**?>**

The high level language is then converted into assembly language either by hand compiling or by using a Compiler and a Linker. When using the latter option, the compiler does this translation by parsing the original source code (in high level language) and converts it to an intermediate format which is in turn translated into the correct machine code suited for the kind of system architecture your computer has. The role of the Linker is to resolve inferences such as subroutine calls within the program. Examples of Compilers are C and C++.Below is an example of an arithmetic instruction in assembly language:

**MOV(AX,4)**

**MOV(BX,5)**

**ADD(AX,BX)**

**OUT(AX,199)**

Afterwards the instructions are written in a register transfer language. A register Transfer language is a symbolic representation of notations used to specify the sequence of micro-operations. An example of a register transfer language to perform addition of two numbers is:

**R2 ← R3**  //Data stored in R3 is transferred into R2

**R5 ← R7**  //Data stored in R7 is transferred into R5

**R2 ← R5 + R2** //Addition is done between the values held in R2 and R5 and the result is stored in R2

This is converted into machine language which is composed of 0’s and 1’s. For example, to perform the above instruction that is in Register Transfer Language, the statement below is used:

**R2 ← R3 ….(i)**

**R5 ← R7 ….(ii)**

**R2 ← R5 + R2 ….(iii)**

The machine language is then used by the control unit to carry out the instructions. A machine language is a pattern of bits encoding machine operations and it is specific to a given processor (a system of instructions and data directly understandable by a computer’s CPU. It is usually in 0’s and 1’s). Following the example given in the register transfer language, we can write a machine language from it:

011 000 010 0111 000 ....(i)

111 000 101 0111 000  ….(ii)

010 101 010 0001 000 …(iii)

The *Control Unit* is the hardware that tells the datapath what to do, in terms of switching, operation selection, data movement between ALU components, etc while the *Processor Unit* is the active part of the computer, which does all the work of data manipulation and decision making. The ALU performs I/O on data stored in the register file, while the Control Unit sends (receives) control signals (resp. data) in conjunction with the register file. An internal bus connects all of the components in the CPU.  Data is transferred between components using this bus and control signals are sent from the CU to other components using control lines.The instruction register feeds the control unit with the control word. (The control word contains groups of bits which indicate the operands, destination, and micro-operations involved in the task for the program to solve). An example of a control word is 111 000 101 0111 000 that is used to transfer the contents of register 7 to register 5. The CU sends control signals to various parts of the CPU.  These signals cause various actions to take place at specified times.  Every instruction consists of a sequence of microsteps, each of which takes place in one clock.  Most instructions require more than one clock to complete their execution.  To cause the execution of an instruction, the CU sends out the appropriate set of control signals during each clock.. The program counter (PC) usually contains the address of the next instruction to be executed. Data to be computed is usually held in the registers which is then loaded onto the multiplexers. The two operands are then fed into the ALU where computations take place. The result is passed to the shifter for any shift operations and the output can also be stored in a register.