Introduction

- The project explains how to use and program a SAP, a simple 8-bit computer architecture that can perform basic operations.
- The SAP has a control panel, a hex keypad, a RAM, a register bank, an ALU, and a control ROM.
- The SAP can be operated in different modes: reset, program loading, full run, and single-step run.
- Modified Simple As Possible Computer(MSAP) is an updated version of SAP architecture that can perform arithmetic and logical operations more precisely.

Program Loading

- The program loading mode allows the user to load instructions and data into the RAM of the SAP.
- There are two ways to load a program: auto-load and manual load.
- Auto load reads a file named Code.txt and automatically loads the program into the RAM.
- Manual load lets the user enter the memory address and data from the hex keypad.

PC Run

- The PC run mode executes the program stored in the RAM of the SAP.
- There are two ways to run a program: full run and single-step run.
- Full run runs the program continuously until it encounters a halt instruction or an error.
- Single-step run runs one instruction at a time and allows the user to observe the changes in the SAP components.

Input Method from Keypad

- The input method from the keypad allows the user to provide input data to the SAP during program execution.
- There are different ways to give input depending on the run mode.
- In manual programming mode, only pressing the key buttons will do the job.
- In single-step run mode, after giving the input, one logic toggle in the single-step clock and one logic toggle in the input ready will complete the process.
- In full run mode, only one logic toggle in input ready will complete the process.

Programming

- The programming section describes how to write a program for the SAP using assembly language.
- The program should be written in a file named Code.txt using hexadecimal numbers and upper-case letters.
- The program should follow the syntax rules for direct and indirect addressing, data allocation, and subroutine functions.
- The program should be written sequentially to the memory address and terminated by HLT and RET instructions.

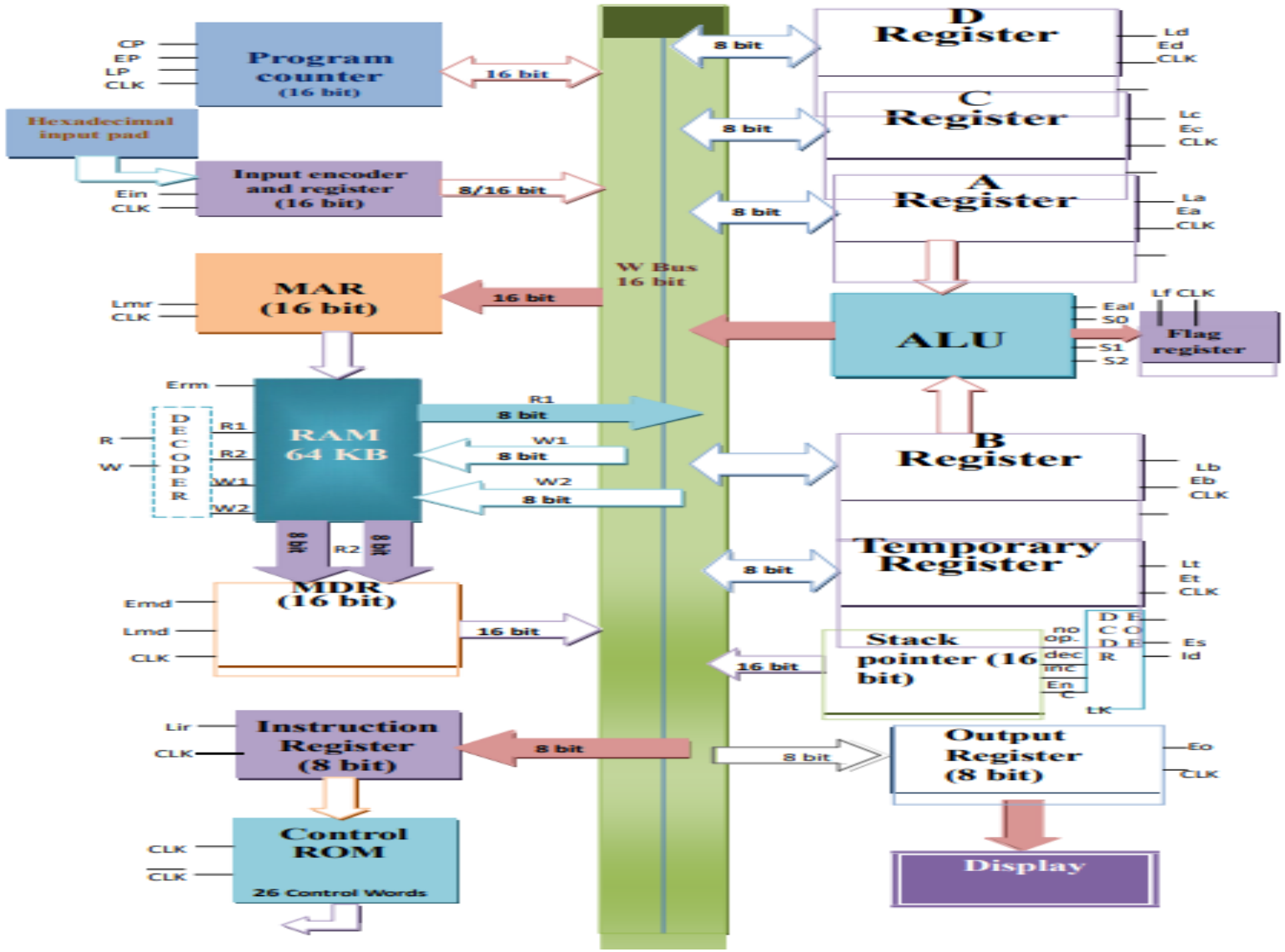
Explanation of control words

- Cp : Increase program counter by 1
- Lp: Loads to program counter
- Ep: Connects program counter to bus
- Ein: Connects input port to bus
- Lmr: Loads to MAR
- Erm: Enables ram, at Erm=1
- RW
- = 00 means data from ram is sent to lower nibble (8 bits) of bus
- =01 means data from ram is sent to MDR
- =10 means data from upper nibble from bus is received by ram
- =11 means data from lower nibble from bus is received by ram

Emd & Lmd:
00--- do nothing
01-- load data to lower nibble of MDR 10-- load data to higher nibble of MDR
11--
enable output of MDR
Lir: Loads to the Instruction register
J: Checks whether the jump condition is satisfied or not (absent in block diagram)
La: Loads to register A
Ea: Sends the data of register A to the bus
Eal: Sends result of ALU to bus
Lf: Updates flag register

S0,S1,S2
001= CMP
010=XOR
011= RCL
100= INC
101= MUL
000/111= no operation (HLT)
Lb: Loads to register B
Eb: Sends the data of register B to bus
Lt: Loads to Temporary register
Et: Sends the data of Temporary register to bus

Es, Id:
00= no operation
01= decrease stack pointer value by 1
10= increase stack pointer value by 1
11= Connects stack pointer register to bus
Eo: Enables output register
Lfb: Loads to flag register
Efb: Sends the data of flag register to bus


CP
EP
LP
CLK
Program counter (16 bit)
16 bit
Hexadecimal input pad
Ein
CLK
Input encoder and register (16 bit)
8/16 bit
W Bus 16 bit
8 bit
D Register
Ld
Ed
CLK
C Register
Lc
Ec
A Register
La
Ea
Lmr
MAR (16 bit)
ALU
Eal
S0
S1
S2
Lf
Flag register
Erm
R
W
DECODER
R1
R2
W1
W2
RAM 64 KB
B Register
Lb
Eb
Temporary Register
Lt
Et
Emd
Lmd
MDR (16 bit)
Stack pointer (16 bit)
no op.
dec
inc
En
Es
ld
Lir
Instruction Register (8 bit)
Output Register (8 bit)
Eo
Control ROM
26 Control Words
Display