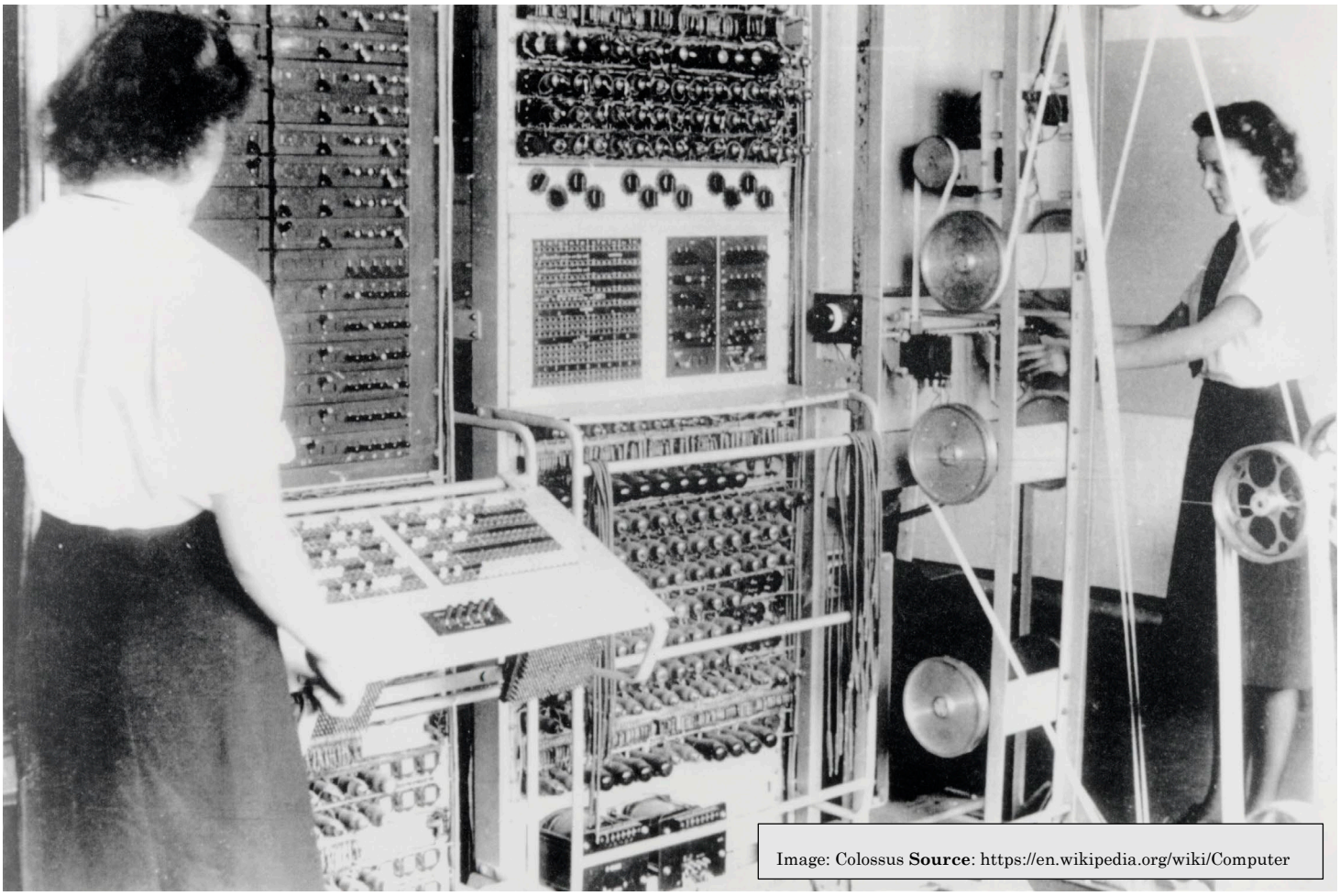


Image: Colossus **Source**: https://en.wikipedia.org/wiki/Computer

DEU Electronic Universal Automatic Reduced Computer (DEUARC) Design

CME 2206 Computer Architecture 2019-2020 Term Project

CME 2206 – LAB PROJECT

DESCRIPTION

You will design a basic computer that called DEUARC (**D**EU **E**lectronic **Uni**versal **A**utomatic **R**educed **C**omputer). DEUARC has 9 registers, 3 memories, arithmetic and logic unit, control unit and bus system.

Quartus II software will be used to design and verify DEUARC. The project is given as a term project and will be implemented in weekly lab sessions. It is advised that you read problem definitions of all of them before actually starting to implement your design, i.e., Common Bus and Registers.

Please submit zipped All Files (don't forget to submit waveform of the results) of the simulations for each lab session.

GENERAL STRUCTURE OF DEUARC

REGISTERS

DEUARC has 9 registers which are *Address Register, Program Counter, Stack Pointer, Input Register, Output Register, Instruction Register* and *3* general purpose registers.

MEMORIES

In DEUARC, there are three memories, which are ***instruction (32x11), data (16x4) and stack (16x5)*** Each has "read enable" signals and "data inputs". *Data and stack memory* also has "write enable input".

COMMON BUS SYSTEM

Common bus system will be responsible for data flow and provide data transfer between register and/or memories.

ARITHMETIC AND LOGIC UNIT

In ALU, arithmetic and logical operations will be held.

CONTROL UNIT

Control unit processes instructions to direct the micro-operations for computer's memories, registers and arithmetic/logic unit. Control unit consists of decoders and a number of control logic gates. It should produce operation signals and time periods for fetching, decoding and executing the instructions.

PROJECT TIME TABLE

INTRODUCTION COMPUTER ARCHITECTURE LABORATORY AND PROJECT – 12.02.2020

Introduction

REVIEW LOGIC DESIGN AND QUARTUS – 19.02.2020

Homework and quiz

MEMORY-DESIGN - 26.02.2020

Lab Study

COMMON BUS SYSTEM – 04.03.2020

Lab Study

ARITHMETIC AND LOGIC UNIT – 11.03.2020

Discussion

ASSIGNMENT 1 – ARITHMETIC AND LOGIC UNIT –

Upload and grading

COMMON BUS SYSTEM–

Discussion

ASSIGNMENT 2 – COMMON BUS SYSTEM -

Upload and grading

LAB – CONTROL UNIT AND FUNCTIONS –

Discussion

LAB – CONTROL FUNCTIONS AND MICROOPERATIONS TABLE –

Upload, no grading only discussion.

ASSIGNMENT 3 – CONTROL UNIT –

Upload and grading

ASSIGNMENT 4 – COMBINING ALL UNITS –

Upload and Grading

ASSIGNMENT 1 – ARITHMETIC AND LOGIC UNIT

You are expected to implement an Arithmetic and Logic Unit (ALU) design that is suitable for your common bus system (assignment-1) and save it as a block diagram ('symbol file') with the name, "ALU" as shown in Figure 1. Test and simulate your implementation by applying supported operations listed in Table 1. The ALU must support following operations that is selected by the input control X[3..0].

X[3..0]	CODE	OPERATION	SYMBOL	DESCRIPTION
0	0000	$R_d \leftarrow R_S \wedge S_2$	AND	R_S AND S_2 (can be R_x or data) and store result in R_d
1	0001	$R_d \leftarrow R_S \vee S_2$	OR	R_S OR S_2 (can be R_x or data) and store result in R_d
2	0010	$R_d \leftarrow R_S + 1$	INC	Increment content of R_S and store result in R_d
3	0011	$R_d \leftarrow R_S / 2$	DBT	Divide content of R_S by 2 and store result in R_d
4	0100	$R_d \leftarrow R_S \times 2$	DBL	Double content of R_S and store result in R_d
5	0101	$R_d \leftarrow R_S + S_2$	ADD	Add R_S to S_2 (can be R_x or data) and store result in R_d

Table 1 - Arithmetic and Logic Operations

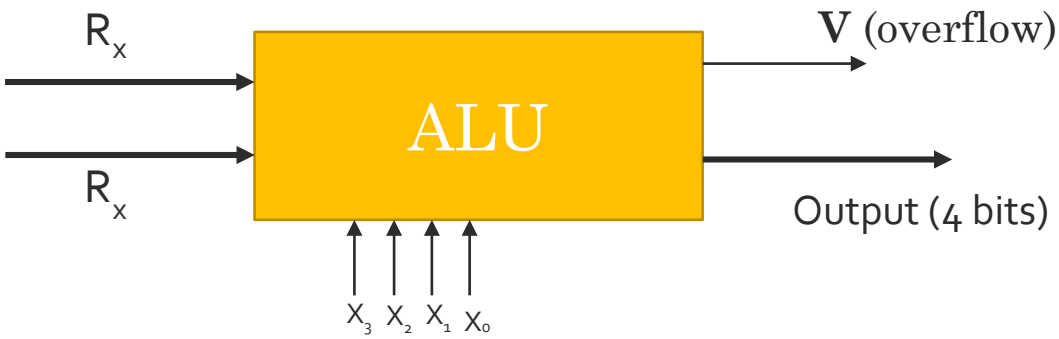


Figure 1 Block Diagram of ALU

ASSIGNMENT 2 – COMMON BUS SYSTEM

You are expected to implement the common bus architecture designed in Quartus II and save it as a block diagram ('symbol file') with the name, "Common_Bus_System". Then test and simulate your implementation by loading (transferring) data from Instruction memory and Data memory to registers and applying INR and CLR operations on them.

REGISTERS

DEUARC has 9 registers which are *Address Register, Program Counter, Stack Pointer, Input Register, Output Register, Instruction Register* and *3* general purpose registers.

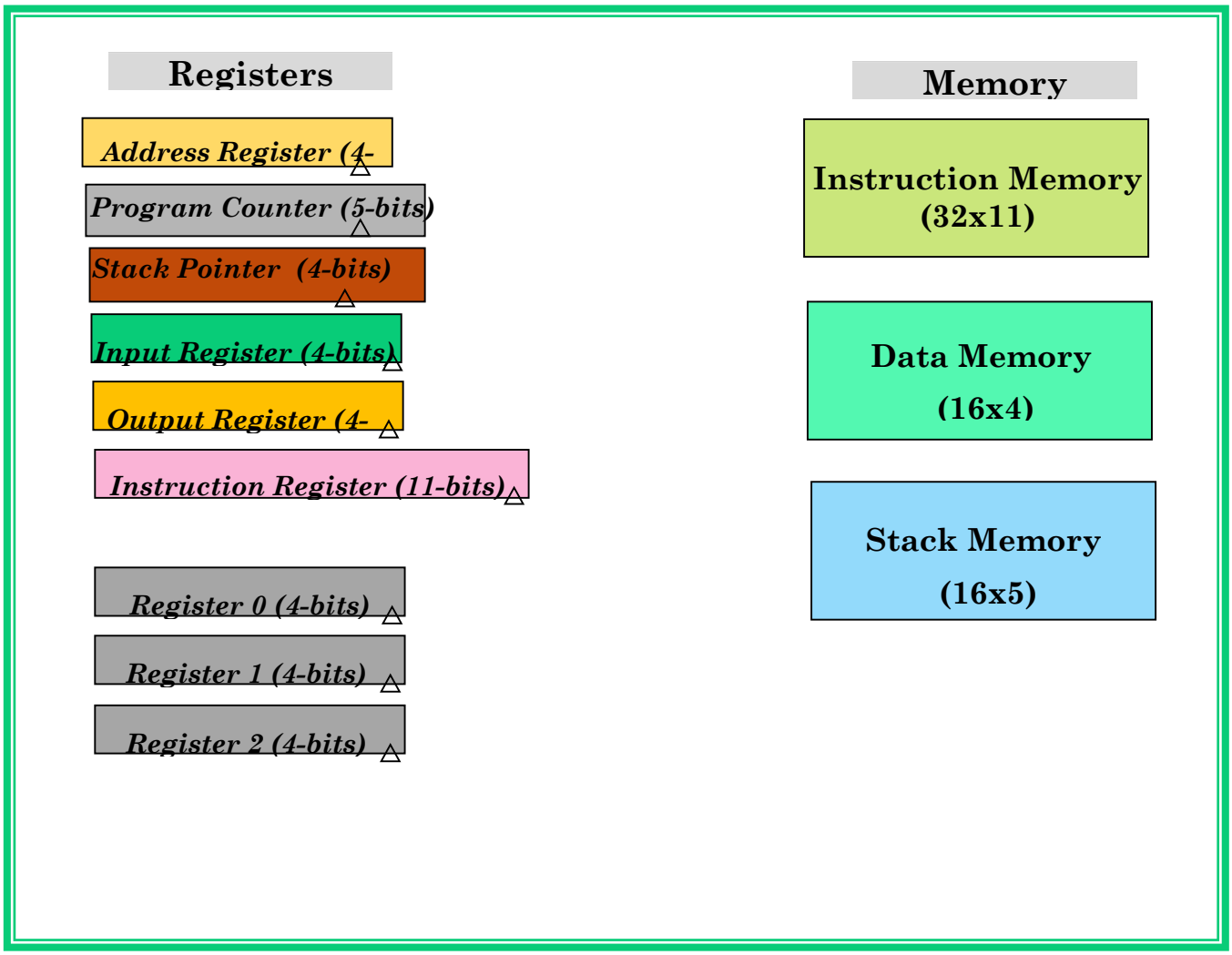
Register Symbol	***Register name***	***Number of bits***	***Function***
AR	Address Register	4	Holds address for data memory. It can be cleared with related control signal.
PC	Program Counter	5	Holds address of instruction. It can be cleared and increase by one with related control signals.
IR	Instruction Register	11	Holds instruction code
SP	Stack Pointer	4	Holds address of the top of the stack memory. It can be cleared, increased and decreased by using related control signals.
InpR	Input Register	4	Holds input data
OutR	Output Register	4	Holds output data
R0, R1, R2	General Purposed Registers	4	Holds operands and other data

Table 2 - List of Registers for DEUARC

MEMORY

In DEUARC there is three memories which are ***instruction, data and stack*** memories. Each has read enable and data inputs. Data and stack memory have also write enable input.

1. Instruction Memory (32x11)
2. Data Memory (16x4)
3. Stack Memory (16x5)


Registers
Address Register (4-
Program Counter (5-bits)
Stack Pointer (4-bits)
Input Register (4-bits)
Output Register (4-
Instruction Register (11-bits)
Register 0 (4-bits)
Register 1 (4-bits)
Register 2 (4-bits)
Memory
Instruction Memory (32x11)
Data Memory (16x4)
Stack Memory (16x5)

ASSIGNMENT 3 – CONTROL UNIT

DEUARC has three instruction code formats as shown in the *Table 3 - DEUARC Instruction Set* The type of the instruction recognized by the computer control unit using four-bits opcodes. You should generate a list for the control function and micro operations of DEUARC (as Table 5.6 of Mano's Basic Computer) before designing control unit. Control unit includes logical designs to control registers, memories, common bus and ALU. Figure 3 and Figure 4 show general view of control unit design and DEUARC respectively. **But they haven't to be completed or correct, so you may add or change signals, components etc. in your designs.**

Table 3 - DEUARC Instruction Set

Symbol	Opcode	Description
HLT	1110	Halt the computer
Arithmetic and Logic Operations		Q(1 bit) | Opcode (4 bits) | R_d (2 bits) | S1 (2 bits) | S2 (2 bits)
AND	0000	AND contents of S1 and S2 and store result in Rd
OR	0001	OR contents of S1 and S2 and store result in Rd
INC	0010	Increase content of S1 and store result in Rd
DBT	0011	Divide content of S1 by 2 and store result to Rd
DBL	0100	Double content of S1 and store result in Rd
ADD	0101	Add content of S1 and S2 and store result in Rd
Data Transfer		Q(1 bit) | Opcode (4 bits) | R_d (2 bits) | S1 (2 bits) | S2 (2 bits)
LD	0110	read the memory content of address *S1S2* and load it into R_d, if Q=0 read the data *S1S2* and load it into R_d, if Q=1
TSF	0111	transfer data from register that is indicated by S1 (R_{S1}) into R_d.
ST	1000	write the content of R_d into the memory of address S1S2
	Registers	00→ R0, 01→ R1, 10→ R2. 11 → INPR **OR** OUTR
Program Control		X(1 bit) | Opcode (4 bits) | X(1 bit) | Address (5 bits)
RET	1001	load the previous PC content from the stack into PC (POP operation of stack memory)
CAL	1010	go to the address of the instruction memory (PUSH operation of stack memory)
JMP	1011	jump to address (5-bits)
CJMP	1100	if V=1 then jump to address (5-bits)
JMR	1101	X (1 bit) | Opcode (4 bits) | XX(2 bits) | Address (4 bits - signed) use *Address* as offset and jump to address relatively

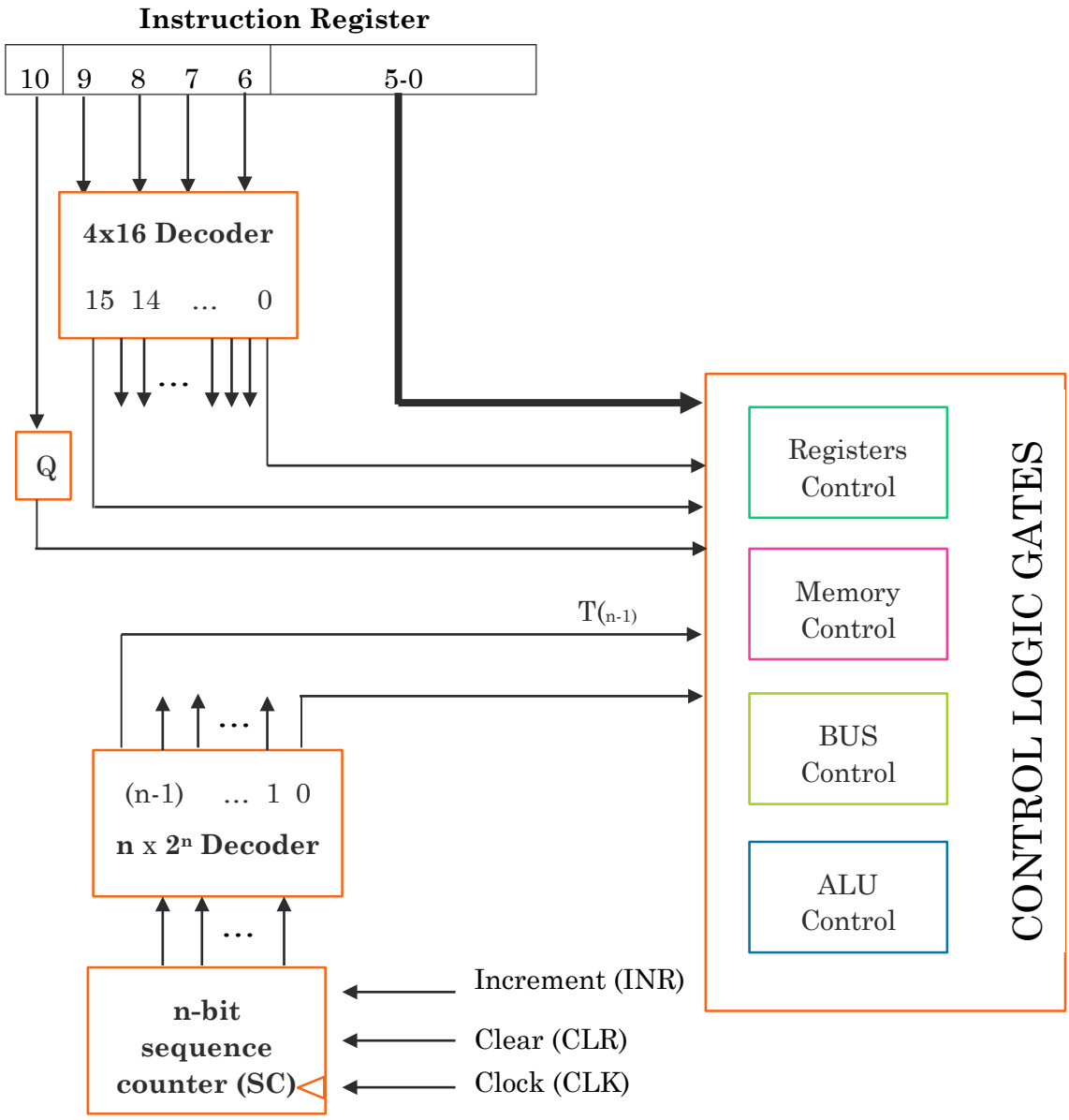


Figure 3 - General structure of Control Unit in DEUARC

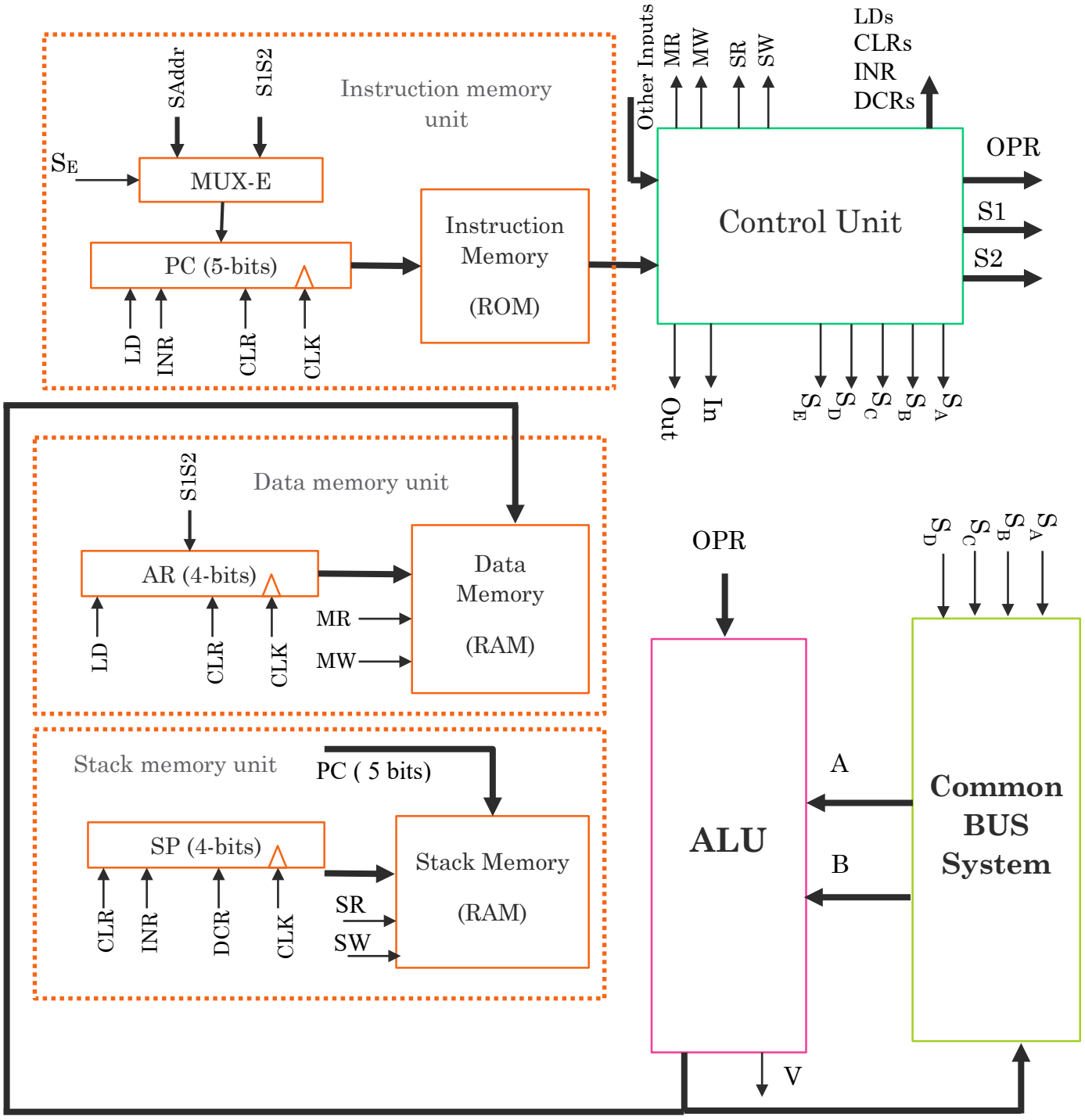


Figure 4 - General view of DEUARC