CME2102 Project Based Learning – IV

DEU Electronic Universal Automatic Reduced Computer (DEUARC) Simulator

Project 2

Teaching Team

Asst. Prof. Dr. Gökhan DALKILIÇ (*Project coordinator*)
Lecturer Şerife YILMAZ (*Project coordinator*)
Assoc. Prof. Dr. Adil ALPKOÇAK

Res. Asst. Dr. Mete Uğur AKDOĞAN
Res. Asst. İlker KALAYCI
Res. Asst. Mustafa BATAR
Res. Asst. Mansur Alp TOÇOĞLU
Res. Asst. Fatih DİCLE
Res. Asst. Özlem YERLİKAYA

DEUARC Simulator Project

In this project, you are expected to design and develop a simulator for a basic computer with reduced instruction set using object oriented programming paradigm. You should use Java programming language and Swing widget toolkit will to implement. The problem domain description for the DEUARC (DEU Electronic Universal Automatic Reduced Computer) is given below.

DEUARC has 9 registers and 3 memory segments (*Figure 1)*. Registers are *Address Register, Program Counter, Stack Pointer, Input Register, Output Register, Instruction Register* and *3* general purpose registers. Each register has data, load, clear and clock inputs. Additionally, program counter has increment input and stack pointer has increment and decrement inputs. Memory has three segments which are *instruction, data* and *stack memory segments*. Each has read enable and data inputs. Data and stack memory segments have write enable input as an extra. Also DEUARC has control unit and ALU (arithmetic logic unit). Control unit processes instructions to direct the micro-operations for computer's memory, registers and arithmetic/logic unit. ALU operates arithmetic and logic operations such as ADD and AND.

DEUARC simulator converts the assembly code to machine code and simulates the program execution phases. It runs the program step by step while showing the phases of instruction cycle (fetching, decoding, execution). DEUARC simulator reads and parses the assembly code, then shows its label table and memory content table. The simulator shows contents of the registers, memory segments, computer operations and their micro operations. It can simulate all operations that DEUARC supports (Table 1). It provides switching between binary / hexadecimal / decimal numbers and exporting *hex* or *mif* file of the machine code (HEX code or binary code).

The input files (*asm* or *basm* file) include assembly (symbolic) codes. The assembly language of the basic computer is defined by a set of rules. Data is indicated by '#' and address is indicated by '@' characters. An example for assembly code is given in the *Code 1*. Each line of the language has three columns called fields.

1. The *label field* may be empty or it may specify a symbolic address. It is followed by a colon (:).
2. The *instruction field* specifies a machine instruction (*Table 1*) or a pseudo-instruction (`ORG, END, DEC N, HEX N`).
3. The *comment field* may be empty or it may include a comment after a sharp sign (%).

The format of a line is as follow:

```
[label]:   instruction [%comment]
```

The weekly plan you will need to follow in the project is given in *Table 2*. At the end of the project, you should make end user document such as tutorial, manual etc. and should make a demonstration in exhibition.

```
     ORG C 2              %Origin of the instruction memory segment
     LD R0,@A             %Load data in the address A into the register R0
     LD R1,#2             %Load 2 (data) into the register R1
     ADD R2,R0,R1         %ADD R0 and R1, then store the result to R2
     ST R2,@S             %Store the content of R2 to the address S indicates
     HLT
     ORG D 3              %Origin of the data memory segment
A:   DEC 8
S:   HEX 5
     END                 %End of symbolic program
```

Code 1 Assembly Code to Add Two Numbers

Table 1 - Instruction set for DEUARC

Symbol		Description
Operation	***Opcode***	
HLT	1000	Halt the computer
		Q(1 bit) | Opcode (4 bits) | D (2 bits) | S1 (2 bits) | S2 (2 bits)
Arithmetic and Logic Operations		
ADD	0000	Add content of S1 and S2 and store result in D
INC	0001	Increase content of S1 and store result in D
DBL	0010	Double content of S1 and store result in D
DBT	0011	Divide content of S1 by 2 and store result to D
NOT	0100	Complement S1 content and load the result into D
AND	0101	AND contents of S1 and S2 and store result in D
		Q(1 bit) | Opcode (4 bits) | D (2 bits) | S1 (2 bits) | S2 (2 bits)
Data Transfer		
LD	0110	reading the memory content of address *S1S2* and load it into D, if Q=0 reading the data *S1S2* and load it into D, if Q=1
ST	0111	writing the content of D into the memory of address S1S2 if Q=0 writing the content of D into the register S2 if Q=1
TSF	1001	Transfers data from S1 into D.
	Registers	00→ R0, 01→ R1, 10→ R2. If S1 = 11 then INPR, if D=11 then OUTR
		- (1 bit) | Opcode (4 bits) | - (2 bits) | - (1 bit) | Address (5 bits)
Program Control		
CAL	1010	go to the address of the instruction memory (PUSH operation of stack memory)
RET	1011	load the previous PC content from the stack into PC (POP operation of stack memory)
JMP	1100	if Q=0 then jump to address (5-bits) if Q=1 then if V=1 then jump to address (5-bits)
JMR	1101	- (1 bit) | Opcode (4 bits) | - (2 bits) | - (2 bits) | Address (4 bits - signed) Use Address as offset and jump to address relatively
PSH	1110	PUSH operation of stack memory
POP	1111	POP operation of stack memory

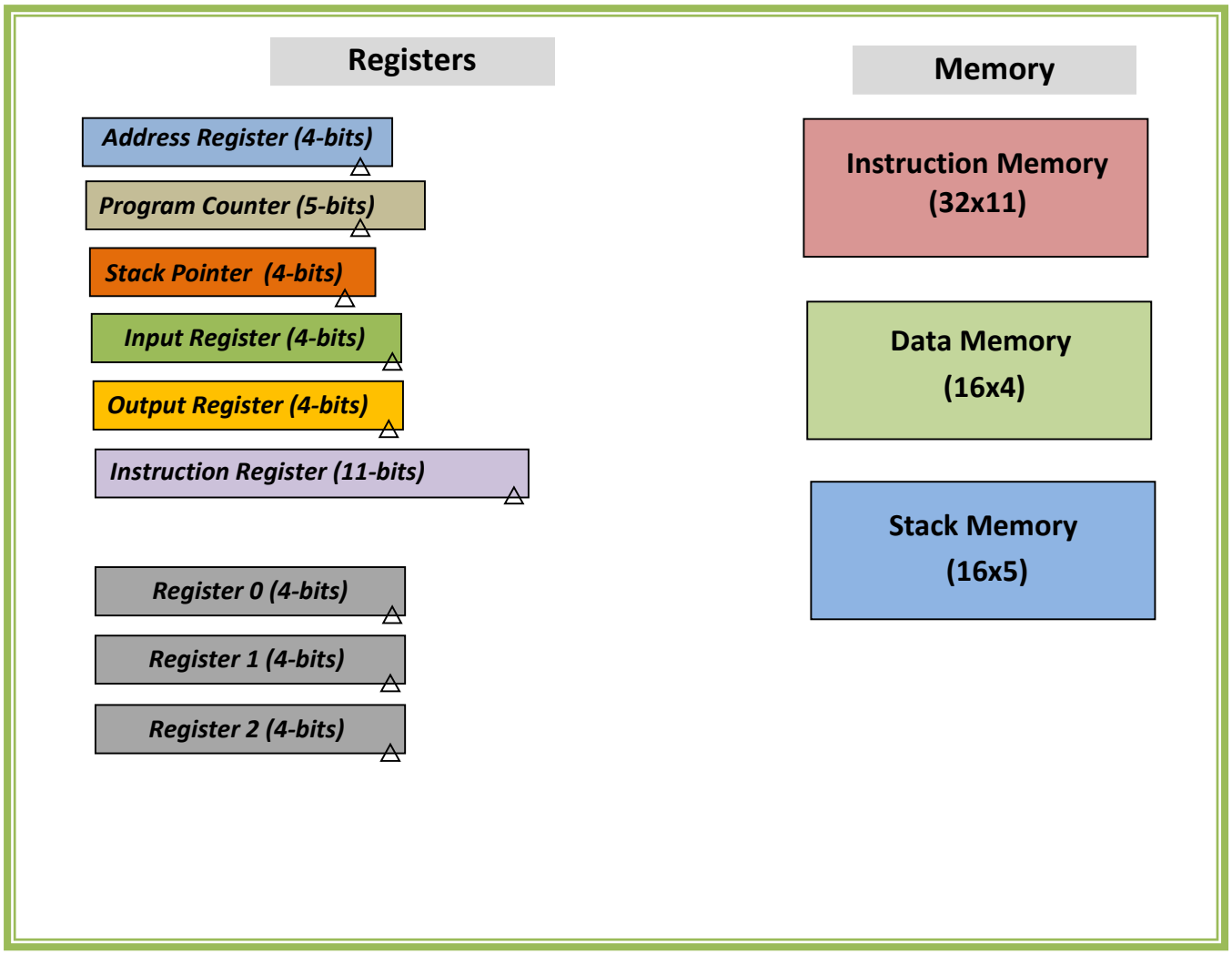

Figure 1 - Registers and Memory of DEUARC