Brett Chow, Aleksandar Aleksandrov

ECEC 355

Project 2

Section 4:

1. (a) For this case, one integer, considering a RISC-V architecture occupies 8 bytes of space.

   The number 2882400001 exceeds the maximum 32-bit integer size. Therefore, the array consists of long integers (64-bits) and one long integer occupies 8 bytes.

   For convenience, we have converted the integers to hexadecimal.

   (b) Data Mapping

0	1	2	3	4	5	6	7
67	**45**	**23**	**01**	**00**	**00**	**00**	**00**
8	9	10	11	12	13	14	15
01	**EF**	**CD**	**AB**	**00**	**00**	**00**	**00**
16	17	18	19	20	21	22	23
3D	**2C**	**1B**	**0A**	**00**	**00**	**00**	**00**

   int arr[3] = {19088743, 2882400001, 169552957}

   19088743 = 0x01234567; 2882400001 = 0xABCDEF01; 169552957 = 0x0A1B2C3D

   Data mapping considering RISC-V Architecture: **LITTLE ENDIAN**

2. Value of x10? Assume x23 = 0

   x8 = 0(x23) → x8 = address of arr[0]

   x9 = 16(x23) → x9 = address of arr[2]

   0 1 2 3 4 5 6 7

   + 0 A 1 B 2 C 3 D

   0 B 3 E 7 1 A 4

   x10 = 0x0B3E71A4

Section 5: Experiment Summary

In this experiment, we create a simulation of a RISC-V Datapath. The code supports the instructions – add, ld, addi, slli, bne. In Core.c every function represents a separate Datapath component. Core.h includes the necessary data structures which hold the information for the instructions and the components. An additional decoding function was made to extract the necessary information from the binary representation of the instruction and convert it to a decimal to be used by the other components. The “tickFunction” simulates the clock cycles and for each clock cycle an instruction is fetched, decoded and then passed through the components.

The ALU component has an implemented shift functionality to support the “slli” function. Passing data to some of the components is determined by the MUX function which is passed as an argument. The dataflow is created by calling the component functions in order and executing based on the control unit values for the corresponding instruction.

Given the instructions:

add x10, x10, x25

ld x9, 0(x10)

addi x22, x22, 1

slli x11, x22, 3

bne x8, x24, -4

Given the values:

x25 = 4

x10 = 4

x22 = 1

x8 = 0

x24 = 0

data_memory = {16, 128, 8, 4}

Output:

x9 = 128

x11 = 16