Laboratory Exercise #5
"The Memory"

Instructions: Redesign the Main Memory based on the memory organization of TRACS (column and row decoding).

1. Develop a new Main Memory function. Refer to the following details:
   a. The memory will be composed of 32 bit x 32 bit chips. Thus, for a memory size of 8 bit x 2048 (16,384 bits or 16Kbit), it requires sixteen (16) 32 x 32 bit memory chips.
   b. The memory chips are organized into two (2) groups of eight (8) chips (see Fig. 1 for memory organization) via row address and column address multiplexing.
   c. Data will be stored via its column and row address and the chip group.

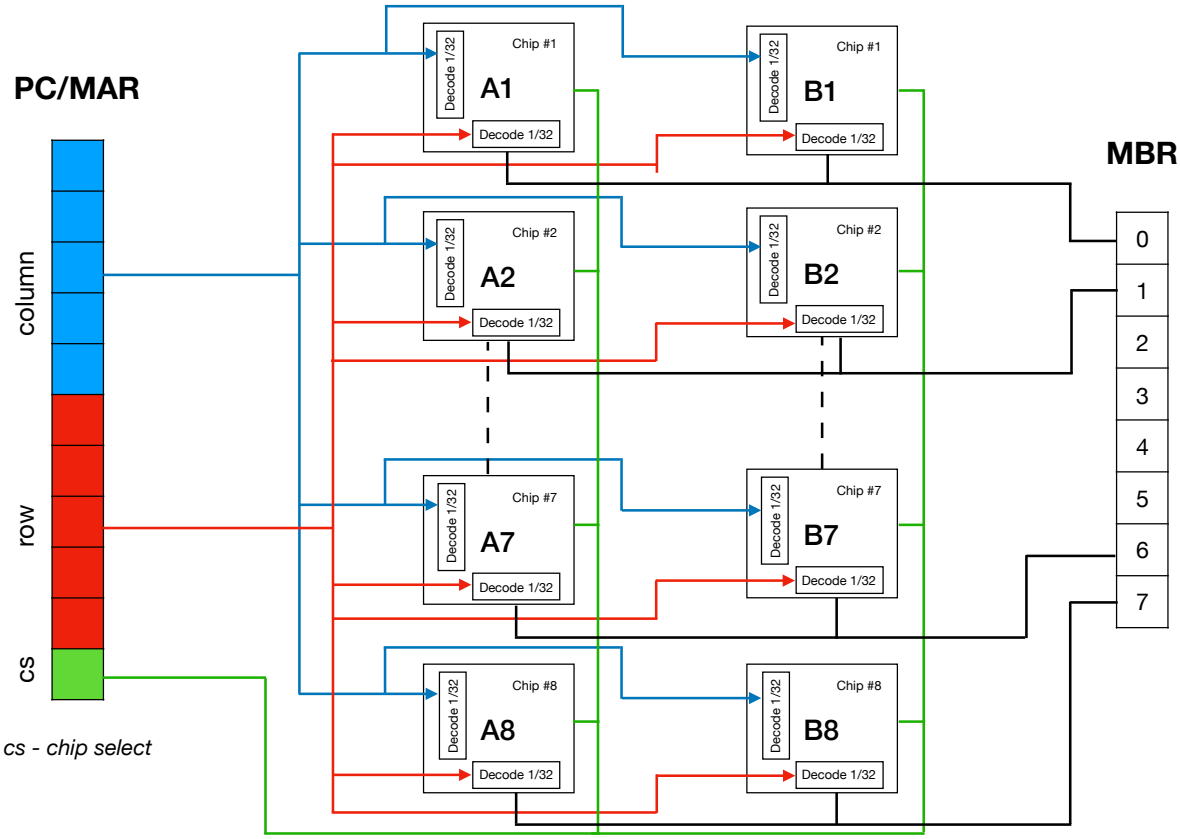

Figure 1. TRACS Memory Organization

   d. Column and row address are 5 bits since the chip is 32 bits x 32 bits.
   e. The chip group selection is controlled by the last bit of PC/MAR which is cs or "chip select".
   f. For example, storing an 8-bit data 0xFA to memory address 0x400 means decoding the row address, column address and chip group: col: 0, row: 0 and cs: 1. Bit 0 of the data is stored on *(0,0)* of chip B1 (since cs = 1). The remaining bits are stored: bit x -> (col, row) of chip A(x-1) if cs = 0, bit x -> (col, row) of chip B(x-1) if cs = 1.

      bit 0 <——> A1(col, row) if cs = 0; B1(col,row) if cs = 1

bit 1 <——> A2(col, row) if cs = 0; B2(col,row) if cs = 1
bit 2 <——> A3(col, row) if cs = 0; B3(col,row) if cs = 1
bit 3 <——> A4(col, row) if cs = 0; B4(col,row) if cs = 1
bit 4 <——> A5(col, row) if cs = 0; B5(col,row) if cs = 1
bit 5 <——> A6(col, row) if cs = 0; B6(col,row) if cs = 1
bit 6 <——> A7(col, row) if cs = 0; B7(col,row) if cs = 1
bit 7 <——> A8(col, row) if cs = 0; B8(col,row) if cs = 1

2. The 32x32 bit chip shall be declared as an array of 32 bit by 32 cells. In this case, the *long* data type will be used since it is 32 bits wide. See the declaration below.

```
…
/* memory chip declaration */
long A1[32], A2[32], A3[32], A4[32], A5[32], A6[32], A7[32], A8[32];      // chip group A
long B1[32], B2[32], B3[32], B4[32], B5[32], B6[32], B7[32], B8[32];      // chip group B
…
```

   The declarations above represents the sixteen (16) chips required to form the 8 x 2048 memory. It is organized into chip groups A & B. The bits are literally stored in the 32 columns and 32 rows of each chip.

3. Decode the address first by breaking it into *col*, *row* and *cs*. Depending on the operation (either read or write), the individual bits of the data shall be extracted or stored based on the row and column address, the chip number and group. Below is a method of decoding the memory address. Note that *col*, *row* and *cs* are local variables within MainMemory( ).

```
…
if(OE)
{
        /* decoding address data */
        col = ADDR & 0x001F;
        row = (ADDR >> 5) & 0x001F;
        cs = ADDR >> 10;
…
```

4. Writing 8-bit data to the memory requires separating each bits and storing in the specified column address and row address and chip group. On the other hand, reading the 8-bit data requires to individual bits from each chip with respect to the column and row address and chip group (see part **f** for details). The objective is to store only one (1) bit in each chip. When writing a bit, the other bits in the row shall not be altered or modified.

5. Since the memory cannot be accessed directly, the initialization of the Main Memory will be done by setting the address bus (ADDR) for the address of the instruction and the data bus (BUS) for the instruction itself. Refer to the format below:

```
…
/* setting the global control signals */
IOM=1, RW=1, OE=1;
/* Format ADDR=<program memory address>; BUS=<instruction>; MainMemory() */
/* Calling MainMemory() writes the instruction to memory */
ADDR=0x000; BUS=0x30; MainMemory();     // write to MBR 0x15
ADDR=0x001; BUS=0x15; MainMemory();
ADDR=0x002; BUS=0x0C; MainMemory();     // write to Main Memory at 0x400
ADDR=0x003; BUS=0x00; MainMemory();
```

   To write the instruction to the Main Memory, the *MainMemory( )* shall be called to write each bit of the instruction to the different memory chip in the group. Global control signals shall be set in order to access the Main Memory.

6. Replace the memory function in Laboratory Exercise #4 with the new one and run the same assembly program. Verify the data being written and read to and from memory. Maintain the code for I/O memory. Save your code as “<LAST NAME>_CPU-MEM.c”.

7. Submit the program in Canvas. Double check if there are no compile errors before submission.

Assessment

Criteria	Outstanding (10 pts)	Competent (8.5 pts)	Marginal (7.5 pts)	Not Acceptable (5 pt)	None (0 pt)
Logic	CPU+Memory logic demonstrated clearly and following exactly the architecture.	CPU+Memory logic demonstrated clearly with slight deviation to the architecture.	CPU+Memory logic demonstrated clearly with several deviations the architecture.	CPU+Memory logic was not demonstrated, did not adhere to the architecture.	
Emulation	Emulation of the CPU+Memory is very close to the actual.	Emulation of the CPU+Memory+IO is slightly close to the actual.	Emulation of the CPU+Memory is very far from the actual.	Emulation of the CPU+Memory is not demonstrated.	
Coding	Coding is neat, systematic, logical and followed accepted coding standards.	Coding is logical and somewhat followed some coding standards.	Coding is logical followed little coding standards.	Coding is messy and did not follow any coding standards.	

Copyright Information

Author: Van B. Patiluna (vbpatiluna@usc.edu.ph)
Contributors: none
Date of Release: March 22, 2021
Version: 1.0

Change log:

Date	Version	Author	Changes
November 8, 2019	1.9	Van B. Patiluna	Original laboratory guide for CpE 415N.
March 22, 2021	1.0	Van B. Patiluna	- Remove the I/O part of the exercises as it will be a separate exercise.