Due: 03/23/2020

Educational Objective: The purpose of this exercise is to investigate and compare the influence of different configurations of cache memory and on-chip and off-chip memory systems on the performance of a microprocessor-based embedded system.

First read through the demo and answer the questions/fill out predictions. Perform experiments to see if your predictions were accurate.

The experiment will be conducted by running simple program loops with different NIOS II configurations. Code is included to use the periodic timer core to measure program loop duration.

Four different system configurations, as shown below, will be considered in this lab. All configurations have a direct-mapped organization.

Name	Processor	I-Cache	D-Cache	Bytes per Cache Line
ConfigB	NiosII/f	512 Bytes	None	N/A
ConfigC	NiosII/f	512 Bytes	512 Bytes	32 (8 words)
ConfigD	NiosII/f	512 Bytes	512 Bytes	16 (4 words)
ConfigE	NiosII/f	512 Bytes	512 Bytes	4 (1 word)

Setup:

1. Download the CacheSupportFiles.zip file from MyCourses and unzip in a working directory.
2. Open the NIOS II software build Tools for Eclipse and set the workspace to the newly created directory.
3. In the NIOS II Software Build Tools for Eclipse, create a new NIOS II application and BSP from template. Make sure that it uses the **ConfigB.sopcinfo** file. Name the App **ConfigB_App** and choose **blank project** for the template.
4. Right click on the BSP folder and choose **NIOS II > BSP Editor**
    a. Choose **none** for system clock timer and **timer_0** for timestamp timer
    b. Click on Generate and then exit
    c. Copy the **system.h** file from the bsp folder to the app folder
5. Repeat steps 3 and 4 for configC, configD and configE. It is important to choose the correct **sopcinfo** file for each application and to change the timestamp timer.

Experiment 1:

In the following program loop, W[ ] is an array of 8192 integers (32 bit numbers). Assume that this program loop is executed with ConfigE.

```
printf("Enter a new value for jmax: ");
scanf("%d", &jmax);
start_measurement();
y = 0;
for(i = 0; i < 100; i++)
  for(j = 0; j < jmax; j++)
    y += w[j];
stop_measurement();
```

The loop is to be run multiple times with values of jmax ranging from1 to 1000. Each time it is run the execution time will be recorded.

Q1: As jmax increases from 1 towards 1000, at which value of jmax will you see a significant increase in the execution time? (hint: execution time will increase linearly until you reach a certain value of jmax. At that point execution time will increase significantly).

Q2: Why does this happen?

1. From within NIOS II Software Build Tools for Eclipse, open the Quartus programmer by selecting **NIOS II > Quartus Prime Programmer**. Program the DE1-SoC board with the **configE.sof** file.
2. Move the **working_set.c** file from the **sw** directory to the **ConfigE_App** folder.
3. Build the project
4. Right click on **ConfigE_App** and choose **run as > NIOS II hardware**
5. Enter the following values and record the execution time in us.

jmax	time
1000	
500	
250	
125	
62	
31	
16	
8	
4	
2	
1	

What do you observe?

Experiment 2:

- The following two loops are run with ConfigC and ConfigE. W[ ] is an array of 8192 integers (32 bit numbers).

Loop 1: (function 1.1)

```
start_measurement();
for (i = 0; i < 1024; i += 8)
    w[i]++;
stop_measurement();
```

Loop 2: (function 1.2)

```
start_measurement();
for (i = 0; i < 128; i++)
  w[i]++;
stop_measurement();
```

Q3: Which configuration would you expect to have the shortest execution time for loop 1?

Q4: Why?

Q5: Which configuration would you expect to have the shortest execution time for loop 2?

Q6: Why?.

1. Move the **functions.c** file from the sw directory to **ConfigC_App** and **ConfigE_App.**
2. Remove the **working_set.c** program from the **ConfigE_App**.
3. Program the DE1-SoC board with **ConfigC.sof**
4. Build the project
5. Click on **ConfigC_App** and choose **run as > NIOS II hardware**.
6. Record the results
7. Repeat steps 3 through 5 for **configE**

Configuration name	Block Size	Function 1.1	Function 1.2
ConfigC	32		
ConfigE	4		

8. How does the data collected compare to your predictions?

Experiment 3:

The following two loops are run with ConfigB (no data cache) and ConfigC (data cache with block size of 32 bytes). Both loops have the same number of memory accesses. W[ ] is an array of 8192 integers.

Loop 1: (function 2.7)

```
start_measurement();
y = 0;
for(i = 0; i < 16; i++)
  for(j = 0; j < 128; j++)
    y += w[j];
```

Loop 2: (function 2.8)

```
start_measurement();
y = 0;
for(i = 0; i < 8; i++)
  for(j = 0; j < 256; j++)
    y += w[j];
stop_measurement();
```

Q7: Which loop would have a shorter execution time in ConfigB? _

Q8: Why?

Q9: Which loop would have a shorter execution time in ConfigC?
Q10: Why?

1. Move the **functions2.c** file from the sw directory to **ConfigB_App**, and **ConfigC_App**
2. Remove the **functions.c** program from **ConfigC_App**.
3. Program the DE1-SoC board with **configB.sof**
4. Click on **ConfigB_App** and choose **run as > NIOS II hardware**.
5. Record the results
6. Repeat steps 4 through 6 for configC

Configuration	Funct. 2.7	Funct. 2.8
ConfigB		
ConfigC		

What do you observe?

Experiment 4:

Consider the following three functions. Assume that **initMatrix** is always run prior to the other two functions. The variable N and the variable size are both equal to 24 and the configuration is B.

```
void InitMatrix (int matrix[N][N]){
  int i, j;

  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      matrix[i][j] = i+j;
    }
  }
}

int SumByColRow (int matrix[N][N], int size)
{
  int i, j, Sum = 0;

  for (j = 0; j < size; j ++) {
    for (i = 0; i < size; i ++) {
      Sum += matrix[i][j];
    }
  }
  return Sum;
}

int SumByRowCol (int matrix[N][N], int size)
{
  int i, j, Sum = 0;

  for (i = 0; i < size; i ++) {
    for (j = 0; j < size; j ++) {
      Sum += matrix[i][j];
    }
  }
```

```
    return Sum;
}
```

These functions are to be run two times. The first time, the data will be stored in SDRAM and the second time, the data will be stored in On-chip memory.

Q11: For which memory would you expect the shortest execution time?

1. Program the DE1-SoC board with **configB.sof**.
2. Add **matrixadditon.c** to the **ConfigB_App** folder
3. Right click on the **configB_bsp** folder and choose **NIOS II > bsp editor**. Click on the **linker script** tab. Verify that SDRAM is chosen for **rodata, stack** and **heap**. Click on generate and then finish
4. Build the project
5. Click on **ConfigB_App** and choose **run as > NIOS II hardware** and record the results below
6. Repeat steps 3 through 5, substituting On-chip RAM for **rodata**, **stack** and **heap**. You can change the memory by double clicking on it and then selecting from pull-down list.

Function	SDRAM	On-Chip
SumByColRow		
SumByRowCol		

Are your predictions correct?

Demo Submission: Submit a document with your predictions along with your results to the dropbox on mycourses.