DSP Lab #2 Report

Course Name: Advanced DSP Applications

Revision: 0

Date: June 13, 2009

Department: School of Engineering and Information Technology

Program: Integrated Telecommunication and Computer Technologies

Prepared For: Peter Roeser, Professor

Prepared By: Jesse Schwartzentruber

Revision History

Revision	Description of Change	Effective Date
0	Initial Release	June 13, 2009

Hands-On Experiment 5.1

Modified exp5_1.asm:

```
//Experiment 5_1
//This assembly program is used to illustrate the example shown in Example 5.3

#include "defBF537.h"

//Setting up the memory in the L1 data memory bank A
.section L1_data_a ;
.align 4;
.BYTE4 buffa[3] = 0x12345678, 0xabcd1234, 0xaabb1234;      //set the first 3
32-bit word
//JS-MOD(a): change gap from 0x1000 to 0x4000 (4096 double words)
.var array[4093];      //set the next 4093 32-bit locations to 0y
.BYTE4 buffa1[2] = 0x00000000, 0x01012020;
.global buffa;
.global buffa1;

//Setting up the memory in the L1 data memory bank B
.section L1_data_b ;
.align 4;
//starting address 0xFF90 2000
.BYTE4 buffb[2] = 0x11223344, 0x55667788;
.global buffb;

.section program;
.global _main ;
_main :
//initialize the pointer and index registers according to Fig.5.5
P0.L = buffa;
P0.H = buffa;
P1.L = buffa1;
P1.H = buffa1;
I0.L = buffa1;
I0.H = buffa1;
M0 = 2;
P2 = 4;
SP.L = buffb+0x4;
SP.H = buffb;

//(a) 32-bit load and store
R0 = [P0];
[P1] = R0;

//(b) 16-bit and 8-bit load
R0 = w[P0](z);
R0 = b[P0](x);

//(c) post increment
R0 = [P0++];
R0 = w[P0++](z);
R0 = b[P0++](z);

//(d) stack pointer predecrement
```

```
[--SP] = R0;            //pre-decrement
R1 = [SP++];            //post-increment

//(e) modify pointer register
P0.L = buffa;
P0.H = buffa;
R1 = [P0+0x08];
R0 = [P0++P2];
R1 = [I0++M0];

// JS-MOD(b): 8 and 16 bit stores
b[P0] = R0;
w[P0] = R0;

// JS-MOD(c): 8 and 16 bit access with I0
//R1 = w[I0++M0]; // illegal (can't do bit extend or post-inc on 16b I0 access)
R1.L = w[I0];
I0 += M0;
//R1 = b[I0++M0]; // illegal (can't do 8 bit access with I0)
P0 = I0;
R1 = b[P0](z);
I0 += M0;

// JS-MOD(d): modify P0 with M0
//R0 = [P0+M0]; // illegal (can't use M reg to modify P reg)
I0 = P0;
I0 += M0;
R0 = [I0];

idle;

_main.end:
```

Hands-On Experiment 5.2

Modified exp5_2.asm:

```
// Exp5_2
// Circular Buffer example as illustrated in Example 5.4

#include <defBF537.h>

.section L1_data_a ;    //Data section begins at L1 bank A memory
.align 4;               //Align data to 4 byte boundary
.byte4 buff[11] = 1,2,3,4,5,6,7,8,9,10,11;     //Each data is 4 bytes
.global buff;           //define buff as global variable

.global _main;
.section program ;      //Program section starts from L1 instruction memory

_main:

      R0 = 0; R1 = 0; R2 = 0; R3 = 0; R4 = 0; R5 = 0;//clear data registers to
0
      L0 = 44;          //length of circular buffer = 11*4bytes
      M0 = 16;          //set modifier register to 4 location (4*4bytes)
```

```
      I0.L = buff;            //set index register to start at buff
      I0.H = buff;

      B0 = I0;          //set base register to start at buff

      R0 = [I0++M0];    //access first element in buff
      R1 = [I0++M0];    //access fifth element in buff
      R2 = [I0++M0];    //access ninth element in buff
      R3 = [I0++M0];    //access second element in buff
      R4 = [I0++M0];    //access sixth element in buff
      R5 = [I0++M0];    //access tenth element in buff

      // break here to observe first 6 accesses
      R0 = [I0++M0];    //access third element in buff
      R1 = [I0++M0];    //access seventh element in buff
      R2 = [I0++M0];    //access eleventh element in buff
      R3 = [I0++M0];    //access fourth element in buff
      R4 = [I0++M0];    //access eighth element in buff
      R5 = [I0++M0];    //access first element in buff

      // break here to observe last 6 accesses
here: jump here;        //remains in this loop when program completes

_main.end:
```

Modified exp5_2a.asm:

```
// Exp5_2
// Circular Buffer example as illustrated in Example 5.4

#include <defBF537.h>

.section L1_data_a ;    //Data section begins at L1 bank A memory
.align 2;               //Align data to 2 byte boundary
.byte2 buff[11] = 1,2,3,4,5,6,7,8,9,10,11;     //Each data is 2 bytes
.global buff;           //define buff as global variable

.global _main;
.section program ;      //Program section starts from L1 instruction memory

_main:

      R0 = 0; R1 = 0; R2 = 0; R3 = 0; R4 = 0; R5 = 0;//clear data registers to
0
      L0 = 22;          //length of circular buffer = 11*2bytes
      M0 = 8;           //set modifier register to 4 location (4*2bytes)

      I0.L = buff;      //set index register to start at buff
      I0.H = buff;

      B0 = I0;          //set base register to start at buff

      R0.L = W[I0];     //access first element in buff
      I0 += M0;
      R1.L = W[I0];     //access fifth element in buff
      I0 += M0;
```

```
      R2.L = W[I0];     //access ninth element in buff
      I0 += M0;
      R3.L = W[I0];     //access second element in buff
      I0 += M0;
      R4.L = W[I0];     //access sixth element in buff
      I0 += M0;
      R5.L = W[I0];     //access tenth element in buff
      I0 += M0;

      // break here to observe first 6 accesses
      R0.L = W[I0];     //access third element in buff
      I0 += M0;
      R1.L = W[I0];     //access seventh element in buff
      I0 += M0;
      R2.L = W[I0];     //access eleventh element in buff
      I0 += M0;
      R3.L = W[I0];     //access fourth element in buff
      I0 += M0;
      R4.L = W[I0];     //access eighth element in buf
      I0 += M0;
      R5.L = W[I0];     //access first element in buff
      I0 += M0;

      // break here to observe last 6 accesses
here: jump here;        //remains in this loop when program completes

_main.end:
```