Computer Science 605.411

Module 9 Example Set 3

1. A byte array that contains two 8-bit elements is declared as:
   unsigned char bdata[2];

   The element bdata[0] contains the value 0x2E
   The element bdata[1] contains the value 0x7F

   The beginning memory address for the array is 0x100900B0.
   Indicate in hex, the contents of the memory byte at location 0x100900B0 and at location 0x100900B1 for

   a) a little endian memory system:

Address	Contents
0x100900B0	0x2E
0x100900B1	0x7F

   b) a big endian memory system

Address	Contents
0x100900B0	0x2E
0x100900B1	0x7F

The elements of an array are stored adjacent to each other in memory beginning with the element with the lowest index followed by the element with the next higher index, etc. In this case each element is a single byte, so the memory endianness makes no difference. The endianness determines the order of the bytes within a multi-byte item in memory.

2. An integer array that contains two 32-bit elements is declared as:
   Unsigned integer idata[2];
   The element idata[0] contains the value 0x11223344
   The element idata[1] contains the value 0x05060708

   The beginning memory address for the array is 0x100900C0.
   Indicate in hex, the contents of the memory byte at location 0x100900C2 and at location 0x100900C5 for
   a) a little endian memory system

Address	Contents
0x100900C2	0x22
0x100900C5	0x07

The array elements are stored consecutively beginning with the first element. However, in this case each element contains 4 bytes. With little endian storage order, the low byte within each element is stored at the lowest address, followed by the next higher byte: 0x44, 0x33, 0x 22 and 0x11 for idata[0] and 0x08, 0x07, 0x06 and 0x05 for idata[1].

   b) a big endian memory system

Address	Contents
0x100900C2	0x33
0x100900C5	0x06

With big endian storage order, the high byte within each element is stored at the lowest address, followed by the next lower byte: 0x11, 0x22, 0x33 and 0x44 for idata[0] and 0x05, 0x06, 0x07 and 0x08 for idata[1].

3. Assume that our single-cycle datapath is required to support the MIPS instruction subset and to sustain a theoretical instruction execution rate of 200 MIPS. For the questions below, you may ignore all times except that required for the memory reads or writes.

   a) What would be the maximum cycle time that the memory could have?
   200 MIPS would correspond to one instruction every $1/2 * 10^8 = 2$ ns. The most time consuming instruction would be either a sw or lw because they are the only instructions in the subset that use a memory operand. So two memory operations (a fetch and an operand read or write) would have to be completed in a single clock cycle. Therefore the memory cycle time must be no more than 1ns.

b) What bandwidth, in bytes per second, is required for the CPU-memory bus to support this single-cycle system? That is, how many bytes per second must the bus carry?
Since two 32-bit (i.e., 4-byte) transfers must be performed in 1ns, the bandwidth = 4 billion bytes per second.

c) If the memory could only support a bandwidth of 128 million bytes per second over the 32-bit data bus, what would be the maximum theoretical MIPS rating for the single-cycle system?
Since the data memory transfers at most 4 bytes at a time, the bandwidth of 128 million bytes/sec would correspond to 32 million memory accesses per second. At two memory accesses per instruction, this would correspond to 16 MIPS.

4. The contents of the memory bytes at locations 0x80000 through 0x80003 are as follows:

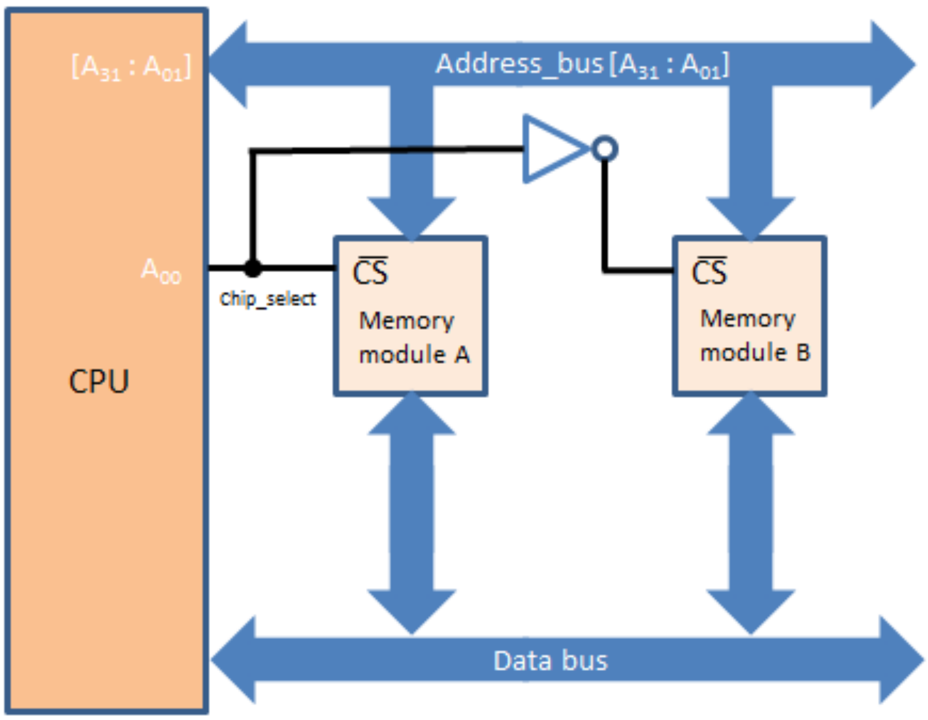
Address	Contents
0x80000	0x47
0x80001	0x87
0x80002	0x76
0x80003	0x34

Show, as an 8-digit hex value, the contents of register $t0 produced by each of the instructions listed below assuming first that the byte addressable system employs big endian storage order and then that it employs little endian storage order. Register $t1 contains the value 0x80000, **lw** is load word, **lh** is load half word, **lhu** is load half word unsigned, **lb** is load byte, and **lbu** is load byte unsigned.

	Big endian	Little endian
lw $t0,0($t1)	0x47877634	0x34768747
lh $t0,0($t1)	0x00004787	0xFFFF8747
lb $t0,3($t1)	0x00000034	0x00000034
lbu $t0,1($t1)	0x00000087	0x00000087
lh $t0,2($t1)	0x00007634	0x00003476
lb $t0,1($t1)	0xFFFFFF87	0xFFFFFF87

The instructions lb and lh sign extend the operand throughout the result register, while lbu and lhu perform zero-extension.

5. The memory system shown below consists of two modules (A and B) each of which contains 2147483648 storage cells. Each storage cell is 16 bits wide. For identification purposes, the cells within each module are numbered starting from 0. This system allows unaligned memory accesses.



CPU
$[A_{31} : A_{01}]$
A_{00}
Address_bus $[A_{31} : A_{01}]$
Chip_select
$\overline{CS}$ Memory module A
$\overline{CS}$ Memory module B
Data bus

The chip select (CS) is active low. That is CS=0 to select a chip.

a) Supply the missing operands for the two li instructions in the sequence that follows such that, when executed, the instruction sequence would store the pattern 0xCAFE into cell 5 within memory module A without affecting any other cells:

```
li      $t1, 0x0A ____
li      $t2, 0xCAFE __
sh      $t2,0($t1)
```

A_{00}, the LSB of the address, selects which module is accessed. The remaining address bits, A_{31} through A_{01} determine which cell within the selected module is accessed. A_{00} must be 0 to select module A and 1 to select module B since the chip select is active low. Hence the address required to access cell 5 within module A is 00000000000000000000000000001010 = 0x0A.

b) Supply the missing operands for the two li instructions in the sequence that follows such that, when executed, the instruction sequence would store the pattern 0xBABE into cell 7 within memory module B without affecting any other cells:

```
li      $t1, 0x0F _
li      $t2, 0xBABE __
sh      $t2,0($t1)
```

The address of cell 7 within module B is 00000000000000000000000000001111 = 0x0F.