Assignment #5
Page 333-338

2. Show how the following values would be stored by byte-addressable machines with 32-bit words, using little endian and then big endian format. Assume each value starts at address 0x10. Draw a diagram of memory for each, placing the appropriate values in the correct (and labeled) memory locations.

a) 0x456789A1 [45|67|89|A1]-->[A1|89|67|45]

3. Fill in the following table to show how the given integers are represented, assuming 16-bits are used to store values and the machine uses 2's complement notation.

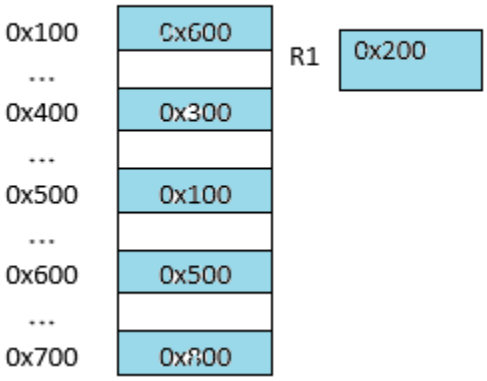
Integer	Binary	Hex	4 Byte Big Endian (hex value as seen in memory)	4 Byte Little Endian (hex value as seen in memory)
-12 1164413345	01000101011001111000100110100001	0x456789A1	[A1|89|67|45]	[45|67|89|A1]

12. Convert the following expressions from infix to reverse Polish (postfix) notation.

c) (W * (X + Y * (U * V))) / (U * (X + Y)) -->W(X+Y(UV))/U(X+Y)-->W(X+YV)/(X+Y)=W+WYV+WX+WV =(1+YV+X+V)=(1+X+V(Y+1))(WU)=WU*VY1+*X1+

22. Suppose we have the instruction Load 500. Given memory and register R1 contain the values below: (view it from the text page 337)

Memory



and assuming R1 is implied in the indexed addressing mode, determine the actual value loaded into the accumulator and fill in the table below:

Mode	Value Loaded into AC
Immediate	
Direct	
Indirect	
Indexed	

Assignment #5
Page 333-338

24. A nonpipelined system takes 100ns to process a task. The same task can be processed in a 5-segment pipeline with a clock cycle of 20ns. Determine the speedup ratio of the pipeline for 100 tasks. What is the theoretical speedup that could be achieved with the pipeline system over a nonpipelined system? This pipeline will take 20ns to process a task at the same speed. The speedup is 5x.

26. Write code to implement the expression: A= (B + C) * (D + E) on 3-, 2-, 1- and 0-address machines. In accordance with programming language practice, computing the expression should not change the values of its operands.

```
add a,b,d
add a,b,e
add a,c,d
add a,c,e
```

```
load a,b
add a,c
load a,d
add a,e
mult
```

```
lda b
add c
lda d
add e
mult
sta a
```

```
push b
push c
add
push d
push e
add
mult
pop a
```