**1. Assume you have a byte-addressable machine that uses 32-bit integers and**

**you are storing the hex value 1234 at address 0:**

1. **Show how this is stored on a big endian machine.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adress 🡪 | 0x0000 | 0x0001 | 0x0002 | 0x0003 |
|  | 00 | 00 | 12 | 34 |

1. **Show how this is stored on a little endian machine.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Adress 🡪 | 0x0000 | 0x0001 | 0x0002 | 0x0003 |
|  | 34 | 12 | 00 | 00 |

**c) If you wanted to increase the hex value to 123456, which byte assignment would be more efficient, big or little endian? Explain your answer.**

Big endian because you can store number after last memory.

**3. Fill in the following table to show how the given integers are represented,**

**assuming that 16 bits are used to store values and the machine uses two’s**

**complement notation.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Integer | Binary | Hex | 4 Byte Big Endian | 4 Byte little Endian |
| 28 | 0000 0000 0001 1100 | 0x001C | 00,1C | 1C,00 |
| 2216 | 0000 1000 1010 1000 | 0x08A8 | 08,A8 | A8,08 |
| -18675 | 1011 0111 0000 1101 | 0xB70D | B7,0D | 0D,B7 |
| -12 |  | 0xFFF4 | FF,F4 | F4,FF |
| 31456 | 0111 1010 1110 0000 | 0x7AE0 | 7A,E0 | E0,7A |

**6. The first two bytes of a 2M × 16 main memory have the following hex values:**

**• Byte 0 is FE**

**• Byte 1 is 01**

**If these bytes hold a 16-bit two’s complement integer, what is its actual**

**decimal value if:**

1. **memory is big endian?**

0FE01000 = 0 1111 1110 0000 0001 0000 0000 0000 0000 0000 = 26633830

1. **memory is little endian?**

000001FE = 0 0000 0000 0000 0000 0000 0000 0001 1111 1110 = 510

**7. What kinds of problems do you think endian-ness can cause if you wished to**

**transfer data from a big endian machine to a little endian machine? Explain.**

It will increases your transfer time because it need to switch all of memory

in main memory.

**10. A computer has 32-bit instructions and 12-bit addresses. Suppose there are**

**250 two-address instructions. How many one-address instructions can be**

**formulated? Explain your answer.**

|  |  |  |
| --- | --- | --- |
| 0000 0000 | xxxx xxxx xxxx | xxxx xxxx xxxx |
|  | | |
| 1111 1001 | xxxx xxxx xxxx | xxxx xxxx xxxx |
| 1111 1010 – escape opcode | |  |
| 1111 1010 | 0000 0000 0000 | xxxx xxxx xxxx |
|  | | |
| 1111 1010 | 1111 1111 1111 | xxxx xxxx xxxx |
| 1111 1011 | 0000 0000 0000 | xxxx xxxx xxxx |
|  | | |
| 1111 1111 | 1111 1111 1111 | xxxx xxxx xxxx |

1111 1010 0000 0000 0000 xxxx xxxx xxxx 🡪 1111 1010 1111 1111 1111 xxxx xxxx xxxx

We have 212 combinations and 1111 1010 🡪 1111 1111 is 5 combinations that mean

212 \* 5 = 20,480 for one-address instructions.

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

**notation.**

1. ***X* × *Y* + *W* × *Z* + *V* × *U***

XYx WZx+ VUx+

1. ***W* × *X* + *W* × (*U* × *V* + *Z*)**

UVx Z+ Wx WXx+

1. **(*W* × (*X* + *Y* × (*U* × *V*)))/(*U* × (*X* + *Y*))**

UVx Yx X +Wx XY+ Ux/

**17. a) In a computer instruction format, the instruction length is 11 bits and the**

**size of an address field is 4 bits. Is it possible to have**

**5 two-address instructions**

**45 one-address instructions**

**32 zero-address instructions**

**using the specified format? Justify your answer.**

|  |  |  |
| --- | --- | --- |
| 000 | xxxx | xxxx |
|  | | |
| 100 | xxxx | xxxx |
| 101 – escape opcode | | |
| 101 | 0000 | xxxx |
|  | | |
| 111 | 1100 | xxxx |
| 111 1101 – escape opcode | | |
| 111 | 1101 | 0000 |
|  | | |
| 111 | 1110 | 1111 |
| 111 1111 – escape opcode | | |

**b) Assume that a computer architect has already designed 6 two-address and**

**24 zero-address instructions using the instruction format above. What is**

**the maximum number of one-address instructions that can be added to the**

**instruction set?**

|  |  |  |
| --- | --- | --- |
| 000 | xxxx | xxxx |
|  | | |
| 101 | xxxx | xxxx |
| 110 – escape opcode | | |
| 110 | 0000 | xxxx |
|  | | |
| 111 | 1110 | xxxx |
| 111 1110 – escape opcode | | |
| 111 | 1110 | 0000 |
|  | | |
| 111 | 1111 | 0111 |
| 111 1111 1– escape opcode | | |

6 two-address instructions escape code is 101

24 zero-address instructions start code is 111 1110

One address instructions is calculate from 110 0000 🡪111 1110 is 30 instructions

**21. Suppose we have the instruction Load 1000. Given that memory and register**

**R1 contain the values below.**

|  |  |  |  |
| --- | --- | --- | --- |
| Memory | |  | R1 0x200 |
| 0x1000 | 0x1400 |  |  |
|  | |  |
| 0x1100 | 0x400 |  |
|  | |  |
| 0x1200 | 0x1000 |  |
|  | |  |
| 0x1300 | 0x1100 |  |
|  | |  |
| 0x1400 | 0x1300 |  |

**and assuming that 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 in to Ac** |
| Immediate | 0x1000 |
| Direct | 0x1400 |
| Indirect | 0x1300 |
| Indexed | 0x1000 |

**27. A digital computer has a memory unit with 24 bits per word. The instruction**

**set consists of 150 different operations. All instructions have an operation**

**code part (opcode) and an address part (allowing for only one address).**

**Each instruction is stored in one word of memory.**

1. **How many bits are needed for the opcode?**

150 operation is need 8 bits to contain.

1. **How many bits are left for the address part of the instruction?**

24 – 8 = 16 bits for address part.

1. **What is the maximum allowable size for memory?**

216 = 65,536

**d) What is the largest unsigned binary number that can be accommodated in**

**one word of memory?**

216 – 1 = 65,535