Michael Gil

**1.1.3 Section Review**

1. Assemblers convert source code from assembly to machine language. Then the linker program combines the individual files created by the assembler into a single executable.

3. The one-to-many relationship between assembly and high level languages means that one high level statement expands into multiple assembly language or machine instructions.

8. type checking is more strict in c / c++ than assembly because c++ does not allow pointers of different types to be assigned to eachother.

12. translate into assembly: X = ( Y \* 4 ) + 3

mov eax, Y

imul eax, 4

mov ebx, 3

add ebx

mov X, eax

**1.2.1 Section Review**

2. Translation is faster because the program can be converted down to machine language by another program written in machine language and then executed directly on the hardware. This is faster than each instruction having to be decoded before execution (Interpretation).

3. When an interpreted program written in language L1 runs, each of its instructions

is decoded and executed by a program written in language L0. **(TRUE)**

10. Assembly statements are translated down to the Instruction set architecture (ISA).

**1.3.9 Section Review**

2. a) 11111000 in decimal is 248.

b) 11001010 in decimal is 202.

c) 11110000 in decimal is 240.

3. a) 00001111 + 00000010 = 00010001

b) 11010101 + 01101011 = 101000000

5. a) 65 needs 7 bits

b) 409 needs 9 bits

c) 16385 needs 15 bits

6. Binary to HEX : a) 35DA , b) CEA3, c) FEDB

7. Hex to Binary

a) A4693FBC = 1010 0100 0110 1001 0011 1111 1011 1100

b) B697C7A1 = 1011 0110 1001 0111 1100 0111 1010 0001

c) 2B3D9461 = 0010 1011 0011 1101 1001 0100 0110 0001

**1.7.1 Short Answer, Questions 1-22 and 25**

1. the MSB is the bit on the far left.
2. 11111000 in decimal is: 248. 11001010 in decimal is : 202. 11110000 in decimal is : 240
3. 00001111 + 00000010 = 00010001 , 11010101 + 01101011 = 101000000
4. 00001101 - 00000111 = 00000110
5. word bits: 16 , doubleword bits: 32 , quadword bits: 64 , double quadword bits: 128
6. 4095 minimum number of bits: 12 , 65534 minimum number of bits: 16 , 42319 minimum number of bits: 16.
7. a) 35DA , b) CEA3, d) FEDB
8. a) 0126F9D4 in binary : 0000 0001 0010 0110 1111 1001 1101 0100 b) 6ACDFA95 to binary: 0110 1010 1100 1101 1111 1010 1001 0101 c) F69BDC2A to binary : 1111 0110 1001 1011 1101 1100 0010 1010
9. a) 3A in unsigned decimal : 58 b) 1BF in unsigned decimal : 447 c) 1001 in unsigned decimal : 4097
10. a) 62 in unsigned decimal : 98 b) 4B3 in unsigned decimal: 1203 c) 29F in unsigned decimal : 671
11. a) -24 in 16-bit hex: E8 b) -331 in 16-bit hex: EB5
12. a) -21 in 16 bit hexadecimal: b) -34 in 16 bit hex: DE
13. a) 6BF9 in decimal: 27641 b) C123 in decimal: 49443
14. a) 4CD2 in decimal: 19666 b) 8230 in decimal : 33328
15. a) 10110101 in decimal : 181 b) 00101010 in decimal : 42 c) 11110000 in decimal : 240
16. a) 10000000 in decimal: 128 b) 11001100 in decimal : 204 c) 10110111 in decimal : 183
17. a) -5 two's complement in binary : 11111011 b) -42 in two's complement binary : 11010110 c) -16 in two's complement binary : 11110000
18. a) -72 in two's complement binary: 10111000 b) -98 in two's complement binary: 10011110
19. a) 6B4 + 3FE = AB2 b) A49 + 6BD = 1106
20. a) 7C4 + 3BE = B82 b) B69 + 7AD = 1316
21. hexadecimal and decimal representations of ASCII character capital B. hex: 42 , decimal: 66
22. capital G in hex and decimal. hex: 47 , dec: 71

25. Truth table for bool function ¬(A v B)

|  |  |
| --- | --- |
| AvB | ￢**AvB** |
| F | T |
| T | F |

**1.4.2 Section Review**

1. (NOT X) OR Y

2. X AND Y

3. T

4. F

5. T

**Algorithm Workbench Question 3 (in C++)**

#include <iostream>

using namespace std;

//function prototype

void showBinary(int, int);

int main()

{

char answer;

do

{

int value1 = 5;

int bin = 0;

cout << "Here is the binary representation of # 5 : ";

showBinary(value1, bin);

cout << endl << endl;

cout << "continue? y/n: ";

cin >> answer;

} while (answer == 'y' || answer =='Y');

return 0;

}

void showBinary(int num1, int num2)

{

do

{

num2 = num1 % 2;

cout << num2;

num1 /= 2;

} while (num1 > 0);

}