CS430
Problem Set #5

Date assigned: 4/1/15
Date due: 4/10/15
Points: 50

Show all work when answering each question for full credit.

1) (15 pts) Consider the following byte-addressable memory where a partial memory dump yields the following:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0F	80	6D	F8	E6	5D	7E	A1	2B	3C	02	00	B1	9B	5C	2A
1	3D	00	9A	1B	D6	B6	0A	A6	F5	E7	9C	1B	B0	E5	A6	7F
2	6A	0C	4C	7E	7C	9D	A2	D7	9B	0F	0A	9F	A2	6C	9C	9B
3	D5	3D	3F	1F	E6	1A	F5	B2	D2	C0	D6	E0	3E	D9	F7	E1
4	4C	8F	A0	5F	0A	D6	5E	2C	F5	3E	9F	E6	0C	8D	9F	F2
5	A9	0F	5F	5D	9D	C5	6D	5E	A1	0A	B7	B6	D5	D4	A7	F9
6	5E	0E	4A	E5	C2	6E	4F	E6	A3	D0	5F	2D	E2	A6	1C	A5
7	0D	E2	F9	C7	3B	2C	5D	A7	6F	A3	0E	C7	E0	D4	F1	4B
8	D8	9F	1F	8E	B9	1E	1B	4E	2F	A6	2A	3A	3B	6B	D8	2E
9	C8	E4	E6	8E	8D	F9	E9	B3	0F	7C	B6	0C	0F	3D	A0	B4
A	F0	2B	5B	A2	A5	6D	B5	F6	8F	C2	1B	0A	9A	1C	B4	4A
B	7F	F2	E2	E3	2A	8A	8F	F9	C4	A3	0D	C7	5D	D8	2F	9F
C	E5	C3	4F	D3	E1	5E	E9	D9	D2	A1	A6	3A	0E	E1	4D	A3
D	D7	B0	D7	A9	A9	5E	A7	D6	4D	D8	A2	9E	7C	7F	F1	4C
E	A4	5D	9E	D9	E4	6F	D2	C1	7B	0D	B9	0B	D0	8B	C0	B7
F			10	00												

Data and addresses are 16-bit and stored using little-endian. A 16-bit processor processes instructions starting at location F0. Assume the instructions look like the following:

F0-F1: Load in AC | Mode
F2-F3: 10 00
F4-F7: Next Instruction

The Mode field specifies an addressing mode and if appropriate indicates a source register that when used is R1 which has a value of 0x0A. Determine the effective address (EA) and the value loaded into the AC for the following address modes:

a) Direct
b) Immediate
c) Indirect
d) Register
e) Register Indirect
f) Displacement

When specifying the EA and value loaded, give your 16-bit result MSb to LSb.

2) (15 pts) Consider the following C program segment:

```
unsigned short int i, j;

if (i < j)
{
  ++i;
}
--j;
```

a) Rewrite the above code in x86 assembly language.

b) In rewriting the above code, you will use the x86 instruction JAE (Jump if above or equal) which executes the jump instruction if CF=0. Explain using clear and complete English sentences why this instruction uses CF=0 in conjunction with the relationship i<= j. This should be understandable to a CS150 student with no architecture experience.

3) (10 pts) Expanding opcodes are a compromise between the desire for a rich set of opcodes and the desire for some short opcodes resulting in shorter instructions. In the x86 world, the instruction inc eax would be 1-byte while the instruction mov ax, 1000h would be a 3-byte instruction.

Consider a computer instruction format where the instruction length is 11-bits and the size of an address field is 4-bits.

a) Is it possible to design an instruction format that has:

- 5 two-address instructions
- 40 one-address instructions
- 32 zero-address instructions

Justify your answer.

b) Assuming a computer architect has already designed 4 two-address instructions and 30 zero-address instructions using the above format. What is the maximum number of one-address instructions that can be added to the instruction set?

4) (10 pts) Write a fully documented C program 05endianPUNetID.c that determines the endianess of a machine. Your program is to print either a) BIG ENDIAN MACHINE or b) LITTLE ENDIAN MACHINE. I do not want you even talking about the solution to this problem at a high level as I am interested in how each of you attack this problem. Paste your C program into your solutions document and report on the endianess of the following machines: (a) zeus (b) ada and (c) circe. I have created an account for each of you on circe which we will talk about in class.

Extra Credit (up to 5 pts)

a) (2 pts) Implement strncpy as defined for C.

```
char *STRNCPY (char *pDest, char *pSrc, int max)
{
```

```
  _asm
  {
    ; your code goes here
  }
}
```

Write adequate code to test this function.

b) (3 pts) Implement a binary search on an array of integers. If the value is not found, return -1; otherwise, return the index of where the value is found.

```
int binarySearch (int values[], int low, int high, int target)
{
  _asm
  {
    ; your code goes here
  }
}
```

Write adequate code to test each function. Also, you can only declare temporary variables in the functions STRNCPY and binarySearch. All other code must be assembly. Here is a simple example.

```
#include <iostream>

int largest (int num1, int num2)
{
  int large;
  __asm
  {
          mov eax, num1
          mov large, eax
          cmp eax, num2
          jge arnd
          mov eax, num2
          mov large, eax
    arnd: nop
  }
  return large;
}

using namespace std;

int main (void)
{
  cout << largest (1, 2) << endl;
  cout << largest (2, 1) << endl;
  cout << largest (1, 1) << endl;
  cout << largest (-1, -2) << endl;
  cout << largest (-2, -1) << endl;

  getchar ();
  return EXIT_SUCCESS;
}
```

Note1: Please make sure your problem sets are typed, answered in order, and stapled together. Also, paste in your code solutions fully documented into your solution.

Note2: A hard copy of your Problem Set Solution is due on the instructor's desk by 11:45am on the day the assignment is due. Also, place a copy of this solution 05PUNetID.doc, 05endianPUNetID.c , and your Visual Studio 10 project 05PUNetID (if you do the extra credit) in a folder called 05PUNetID. Then place the folder 05PUNetID in the CS430 Drop Box by 11:45am on the day in which the assignment is due. As for the endian problem, place a copy of your solution on circe in a directory /home/youruserid/Documents/CS430 with the name 05endianPUNetID.c. This way I can run your program on circe and any other machine.