CS1037A 2019

Lab 2

The purpose of this lab is to demonstrate memory maps and to display actual memory locations of data in computer memory.

PREPERATION:

The Teaching Assistant will discuss two new topics germane to this lab.

1.) hexadecimal numbering

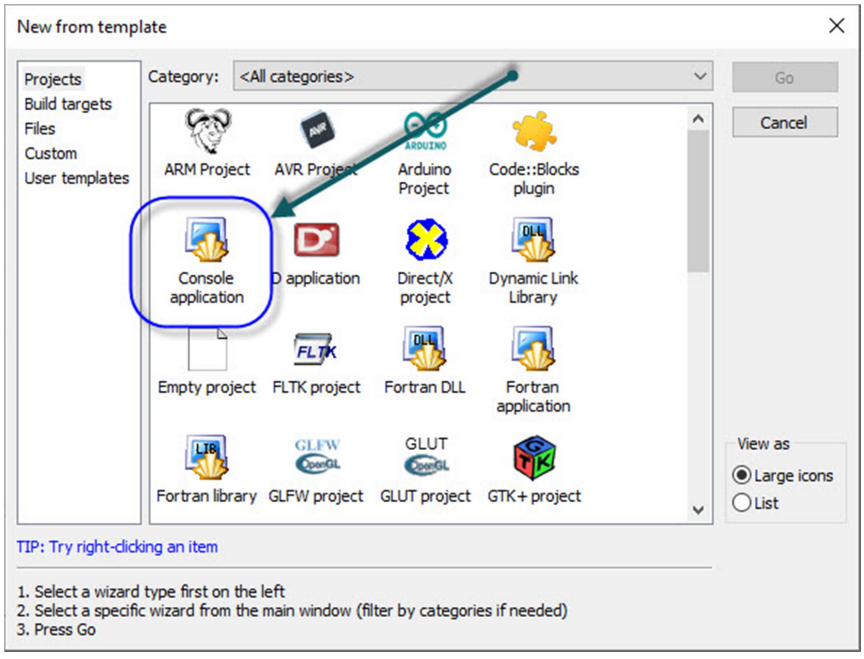
2. C function: sizeof()

LAB 02:

Create a new project in Code::Blocks.

File->New->Project.

1.) Select the <Console Application> option from the [New from Template] screen.
2.) Select the C (NOT the C++) language
3.) Name the Project Title: Lab02
4.) Use the default compiler by clinking on the <Finish> button.



Under [Sources] in the Project window of Code::Blocks, open the main.c file.

Type in the following code EXACTLY as shown:

```c
/* CS1037a 2019 */
/* Lab 02 */
/* Prof Magguilli (your name HERE)*/
/* August 17, 2019 */

#include <stdio.h>
#include <stdlib.h>

  void printBits(size_t const size, void const * const ptr);

int main()
{

char a;
int b;
float c;
double d;

a = 7;        /* 1 byte */
b = -13;      /* 4 bytes */
c = 0.1;      /* 4 bytes */
d = 42.5;     /* 8 bytes */

printf("\n-----------------------------------------\n");

printf("LABEL   -  ADDRESS(hex)   ADDRESS (dec) [S - E]   -     VALUE     -    BINARY\n");

printf("A  -         ");
printf("%p   -    ",&a);
printf("%d  -  %d          ",(int)&a,(int)&a + sizeof(a)-1);
printf("%d  -        ",a);
printBits(sizeof(a), &a);
printf("\n-----------------------------------------\n");

printf("B  -         ");
printf("%p   -    ",&b);
printf("%d  -  %d          ",(int)&b,(int)&b + sizeof(b)-1);
printf("%d  -        ",b);
printBits(sizeof(b), &b);
printf("\n-----------------------------------------\n");

printf("C  -         ");
printf("%p   -    ",&c);
printf("%d  -  %d          ",(int)&c,(int)&c + sizeof(c)-1);
printf("%f  -        ",c);
printBits(sizeof(c), &c);
printf("\n-----------------------------------------\n");

printf("D  -         ");
printf("%p   -    ",&d);
printf("%d  -  %d          ",(int)&d,(int)&d + sizeof(d)-1);
printf("%lf  -        ",d);
printBits(sizeof(d), &d);
printf("\n-----------------------------------------\n");
    return 0;
}
```

In the same file (i.e. the same window in Code::Blocks) type the following code immediately beneath the code you have just entered:

Type in the following code EXACTLY as shown:

```
//assumes little endian
void printBits(size_t const size, void const * const ptr)
{
    unsigned char *b = (unsigned char*) ptr;
    unsigned char byte;
    int i, j;

    for (i=size-1;i>=0;i--)
    {
        for (j=7;j>=0;j--)
        {
            byte = (b[i] >> j) & 1;
            printf("%u", byte);
        }
        printf(" ");
    }
    puts("");
}
```

STANDARDS:
You must add the comments in the code as shown.
This will start you out with the good coding practice of commenting your code.

COMPILE:
Once you have entered the code, compile the code.
You must achieve:

```
Process terminated with status 0 (0 minute(s), 1 second(s))
0 error(s), 0 warning(s) (0 minute(s), 1 second(s))
```

by removing any typographical errors.

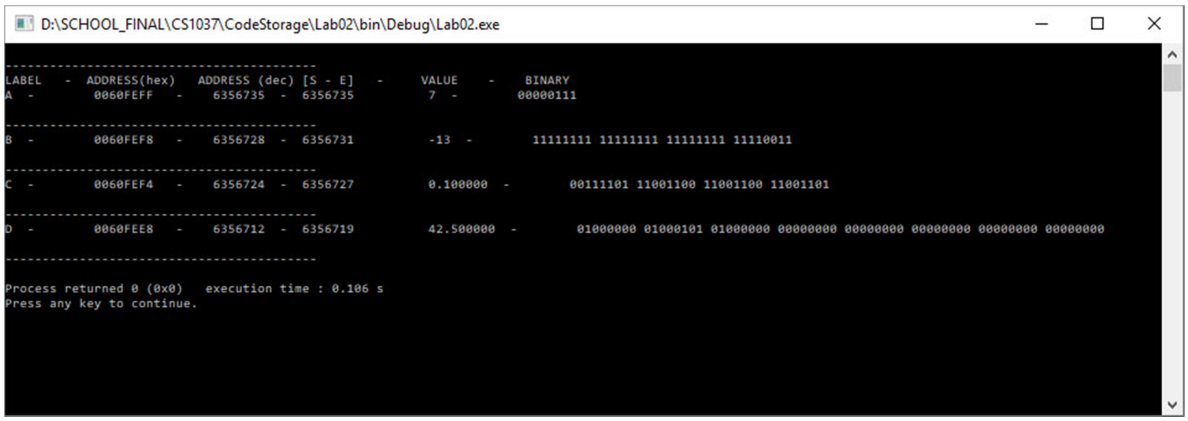
RUN:
Run the code to ensure that it works.

TRACK:
Map out the memory using the following table.

Label	Address	Value	Binary
	…		

EXAMPLE OF EXPECTED RESULTS:

The console screen will look similar to:



Based on the results above, a small example of your memory map will have the following values:

Label	Address	Value	Binary
A	6356735	7	0000 0111
B	6356728	-13	1111 1111
	6356729		1111 1111
	6356730		1111 1111
	6356731		1111 0011
C	6356724		0011 1110
	etc		etc

CHANGE THE ORDER:

Change your code so the order of the variable declaration and the values of the variable definitions are changed to the following:

```
10
11  int main()
12  {
13
14  float c;
15  char a;
16  double d;
17  int b;
18
19  a = 42;          /* 1 byte */
20  b = -83273;       /* 4 bytes */
21  c = 154.6;       /* 4 bytes */
22  d = 561232019;      /* 8 bytes */
23
```

TRACK:

Recompile the and run the program. Map out the new memory allocation and compare tables.

Label	Address	Value	Binary
	…		

FINALLY
Change the order of the variable definitions above to:

```
10
11  int main()
12  {
13
14  float c;
15  char a;
16  double d;
17  int b;
18
19  d = 561232019;      /* 8 bytes */
20  c = 154.6;        /* 4 bytes */
21  b = -83273;        /* 4 bytes */
22  a = 42;          /* 1 byte */
23
```

Compile and run the program again.
Did the memory locations change?
Why (or why not).
We will discuss the answer in class.

FINISH:
Show your work to the TA and make sure your attendance is recorded.