Phys4051: C Lecture 4 & 5

Review:
- Solution to if-statements

New:
- Hardware Examples: Blinking LED
- Switch Statement
- Loop and Bitwise Operator Examples
- Arrays and Array Declaration
- Multidimensional Arrays

Hardware Example 1: Blinking LED (1)

Problem:
An LED is connected to D3 of the databus through the prototype board.

Write a C-function that changes the LED's state without disturbing any of the other bits on the databus.

Hardware Example 1: Blinking LED (2)

Tasks:
1. Read the state of the (buffer) LED
2. Perform a logic operation to change the state of D3
3. Send out (write) the new state to the LED (buffer)
**Hardware Example 1: Blinking LED (3)**

- **Task 1**: To read one byte of data $x$ from port (address) $adr$, use:
  \[ x = \text{inp}( adr ); \]

- **Example**: \[ x = \text{inp}(0x30A); \]

- **Task 3**: To write one byte of data $x$ to port address $adr$, use:
  \[ \text{outp}( adr, x ); \]

- **Example**: \[ \text{outp}(0x300, x); \]

---

**Hardware Example 1: Blinking LED (4)**

- **Task 2 Method 1**: Masking (Bitwise AND):
  \[
  \begin{align*}
  \text{if}( x \& 8 ) & \quad // \text{check } D_3 \\
  x & = x \& 247; \quad // D_3 = \text{HI}, \text{set it LO} \\
  \text{else} & \quad x = x \mid 8; \quad // \text{D}_3 = \text{LO}, \text{set it HI}
  \end{align*}
  \]

---

**Hardware Example 1: Blinking LED (5)**

- **Task 2 Method 1**: Bitwise XOR:
  \[ x = x \wedge 8; \]
Branching Instructions: Switch Statement Syntax

```c
switch( variable )
{
    case CONSTANT :
        statements ;
        break;
}
```

Repetitions and Loops (1) General

- Loop body
- Infinite loop
- Nested loops

Repetitions & Loops (2) Types of loops in C:

- Pre-execution (checking) loops: (may never execute)
  - for loop
  - while loop

- Post-execution (checking) loop: (executes at least once)
  - do-while loop
Repetitions & Loops (3)  
For Loop Syntax

```plaintext
for( init. cond. ; logic cond. ; increment )  
  {  
    statement(s) ;  
  }  

Example:
  for( x = 34; x < 1104; x++)  
    printf("looping ");  //shortcut notation
```

Repetitions & Loops (4)  
While Loop Syntax

```plaintext
while ( logic condition )  
  {  
    statement(s) ;  
  }  

Example:
  short x = 0;  
  while( x > 5)  
    x --;  //shortcut notation
```

Repetitions & Loops (5)  
Do-While Loop Syntax

```plaintext
do  
  {  
    statement(s) ;  
  }  
while ( logic condition ) ;

Example:
  do  
    x = x + 123;  //shortcut notation  
    while( x < 12345 );
```
Example 2: Bitwise Operators & Loop (1)

```c
void char2bin(unsigned char num)
{
    unsigned char i, mask = 0x80;
    for (i = 1; i < 9; i++)
    {
        if (num & mask)
            printf("1");
        else
            printf("0");
        mask = mask >> 1;
    }
}
```

Example 2: Bitwise Operators & Loop (2)

For the following statements:

```c
unsigned char val = 13;
char2bin(val);
```

what is the corresponding output?

Example 2: Bitwise Operators & Loop (3)

```
val = 13; // 0000 1101

<table>
<thead>
<tr>
<th>i</th>
<th>mask</th>
<th>mask &amp; num</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1000 0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Midterm 96: Pseudo Random Number Generator

(7 Points) The picture on the right is from H&H p. 656. It shows a hardware implementation of a 4 bit pseudo random number generator using a shift register and an XOR gate.

For example, if Q3, Q2, Q1, and Q0 are all HI then after one clock cycle, i.e., after all the registers are shifted right, Q = 0111; the next clock cycles produces, Q = 0011, etc.

You will write a C function called MyRand() that implements its hardware function.

Pseudo Random Number Generator: Truth Table

<table>
<thead>
<tr>
<th>cycle</th>
<th>D_in</th>
<th>Q3</th>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
<th>Q3Q2Q1Q0 (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Pseudo Random Number Generator: Flowchart

```c
short MyRand(short Q) {
    #function in header MyRand(short Q)
    //declare local variables Q0, Q1, Q0, Q0 of type short.
    Read the LSB of Q and store it in Q0.
    Read the second LSB of Q and store it in Q1.
    Left-shift the bit stored in Q0 3 positions; store the result in Q40.
    Left-shift the bit stored in Q1 2 positions; store the result in Q41.
    XOR Q40 with Q41; store the result in Q4.
    Add the bit stored in Q4 to Q.
    Right-shift Q by one position; store the result to Q.
    return(Q);
}
```
Pseudo Random Number Generator: Solution

Same thing:

```c
short MyRand( short Q )
{
    return((Q | ((Q&1) <<4) ^ ((Q&2)<<3)) >>1 );
}
```

Arrays

Definition:
- A collection of variables which are all of the same type

Examples:
```c
short val[ 200 ]; //declaration
val[ 12 ] = 5;    //assignment
```

Arrays: Declaration & Data Types

- Arrays have the same data types as variables, i.e., short, long, float etc.
- Arrays are similar to variables: they can either be declared global or local
**Arrays: Number of Elements**

- If an array $y$ has $n$ elements, then:
  - The "first" element is: $y[0]$
  - The last is: $y[n-1]$.

- Example:
  ```
  short y[4];
  contains the following elements:
  y[0], y[1], y[2] and y[3]
  Note: y[4] is **NOT** part of the array!!
  ```

**Arrays: Memory Allocation**

```
short x = 2;
short y[4];
y[x] = 12345;
```

(The memory addresses are arbitrarily chosen)

**Example 4: Calculate Average**

```
float average( void )
{
    short j, x[200];
    float accum = 0;
    for( j = 0; j < 200; j++)
        x[ j ] = rand(); // assign value
    for( j = 0; j < 200; j++)
        accum = accum + x[ j ];
    return (accum / 200);
}
```
Arrays: Initialization

- In global arrays, all elements are initialized to zero.
- In local arrays, the elements are NOT initialized!

Arrays: Declaration & Initialization

- Various examples to declare and initialize arrays:
  
  ```
  short x[12] = {1, 3, 88};
  double w[] = {3.14, 127.0, 22};
  ```

  Strings:
  ```
  char a_name[10] = "Hello";
  char name[20] = {'H','e','l','l','o'};
  ```

Array Size

- Once declared, it’s very difficult to change the size of an array!
- The number of array elements must be declared with a constant. It can NEVER be specified with a variable!
  
  Example:
  ```
  short x = 200;
  short y[x]; //WRONG!!!!!!!!
  ```
Constants (Pre-Processors)

Constant Declaration Example:

#define ACONSTANT 300.0
#define TRUE 1
#define FALSE 0

Note: NO semicolon at the end!
C-convention: the name of a constant is usually written in uppercase.

Example 5: Constant & Average Program

#define MAX 300
short v, x[ MAX ];
main(){
    float accum = 0, average;
    for(v = 0; v < MAX; v++)
        x[ v ] = rand();
    for(v = 0; v < MAX; v++)
        accum += x[v];
    average = accum / MAX;
}

Example 6:

#define MAX 200
short x, volt[MAX], temp;

/*... Various statements that assign values to array volt[] have been omitted*/
for(x = 0; x < MAX / 2; x++)
    {
        temp = volt[x];
        volt[x] = volt[MAX - x -1];
        volt[MAX - 1 - x] = temp;
    }
Example 7: (Program Segment)

```c
for (y = 0; y < MAX - 1; y++)
  for (x = 0; x < MAX - 1 - y; x++)
    if (volt[x] > volt[x + 1])
      { temp = volt[x];
        volt[x] = volt[x + 1];
        volt[x + 1] = temp;
      }
```

Example 7: Content of Array “volt”

| Array  | Index | y=0, x=0 | y=0, x=MAX-1 | y=1, x=MAX-2 | Finally |
|--------|-------|----------|--------------|--------------|
| 0      |       |          |              |              |         |
| 1      |       |          |              |              |         |
| 2      |       |          |              |              |         |
| 3      |       |          |              |              |         |
| 4      |       |          |              |              |         |
| 5      |       |          |              |              |         |

Arrays: Memory and Size Considerations (1)

- Global arrays are always “static”
- Local arrays are always “dynamic”

- The “static” memory space is usually larger than the space allocated to “dynamic” variables and arrays!
- To make a local array “static”, declare it so explicitly: `static short x[200];`
Arrays: Memory and Size Considerations (2)

* Keep track of array size in Bytes!

```c
char c[10000]; // 10000 bytes
double d[10000]; // 80000 bytes
```

* Beware of the 64 kB boundary in some compilers!

Arrays: Memory and Size Considerations (3)

* When in doubt use the `sizeof` keyword:

```c
char char;
double double;
printf("%ld \n", sizeof( char ) );
printf("%ld \n", sizeof( double ) );
```

* Output:

```
1
8
```

Multi-Dimensional Arrays

* 2D Array Example:

```c
short daysofyear[2][13] = {
    { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 
      30, 31, 30, 31 },
    { 0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 
      30, 31, 30, 31 }
} 
```

* Usage:

```c
days = daysofyear[leapyear][month]
```
**C-Function: Most Basic Form**

Function Declaration: 

Function Header: 

```c
{
}
```

Statement(s): 

```c
;
```

**C-Function: General Form**

Function Declaration: 

Function Header: 

```c
{
}
```

Local Variable Declarations: 

```c
;
```

Statement(s) and other Function Calls: 

```c
;
return( );
```
if Statement

```java
if(logic condition)
{
    Statement(s);
}
```

if-else Statement

```java
if(logic condition)
{
    Statement(s) I;
} else
{
    Statement(s) II;
}
```
for - Loop

for(I. Initial condition; II. Logic condition; III. Increment)
{
    Statement(s);
    
    
}

Example:

for( x = 11 ; x <= 4051 ; x++ )
{
    printf("hello") ;
}

I. Initial Condition

II. Logic condition

TRUE

FALSE

III. Increment

Statement(s)
**while - Loop**

```
while (logic condition)
{
    // Statement(s)
}
```

**do-while - Loop**

```
do
{
    // Statement(s)
}
while (logic condition);
```