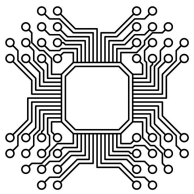
SERIE 5 – PROGRAMMING IN C - REMOTELY

Information systems



Question 1 – *Types*

You are given the following declarations:

```
int n = 10; int p = 4;
long q = 2;
float x = 1.75;
```

Give the type and the value of the following expressions (beware differences with Java!)

1. `n+q`
2. `n+x`
3. `n % p + q`
4. `n < p`
5. `n >= p`
6. `n > q`
7. `q + 3 * (n > p)`
8. `-q && n`
9. `(q-2) && (n-10)`
10. `x * (q==2)`
11. `x * (q=2)`

Question 2 – *Pointers*

Let there be the following code:

```
1  int a, b;
2  int * p, * q;
```

With those declarations, what are the values of `a`, `b`, `*p`?

1.
```
1  a = 3;
2  b = 4;
3  p = &a;
```

2.
```
1  a = 1;
2  b = 2;
3  p = &a;
4  q = p;
5  b = *q;
```

3.
```
1  a = 1;
2  p = &a;
3  b = a;
4  a = 2;
```

4.
```
a = 1;
b = 2;
p = &a;
*p = b;
b = a;
a = *p;
```

5.
```
a = 3;
b = a;
q = &b;
p = q;
```

6.
```
a = 1;
b = 2;
b = a;
p = 0;
```

Question 3 – *First steps in C*

(a) To start writing code in C using MPLAB you need to:

1. Create a new project in MPLAB as usual but instead of the assembler (`mpasm`) choose the `xc8` compiler.
2. Download[1] the configuration source files we prepared for you. Copy them both to the project folder.
3. Right-click on the project and choose *Add existing item* and select the `picebs2` files (c and h).
4. You can now add your main source file by selecting *File* → *New file* (Ctrl + N).
5. Configure the simulator to react like the real PIC. To do so, right-click on the project again and select *Properties* (the last option). Select then the *simulator* option, set the instruction frequency to 2 MHz and then change the *option categories* to *Uart1 IO options*. Enable the *Enable Uart1 IO* option as depicted in Figure 1. You are now all set!

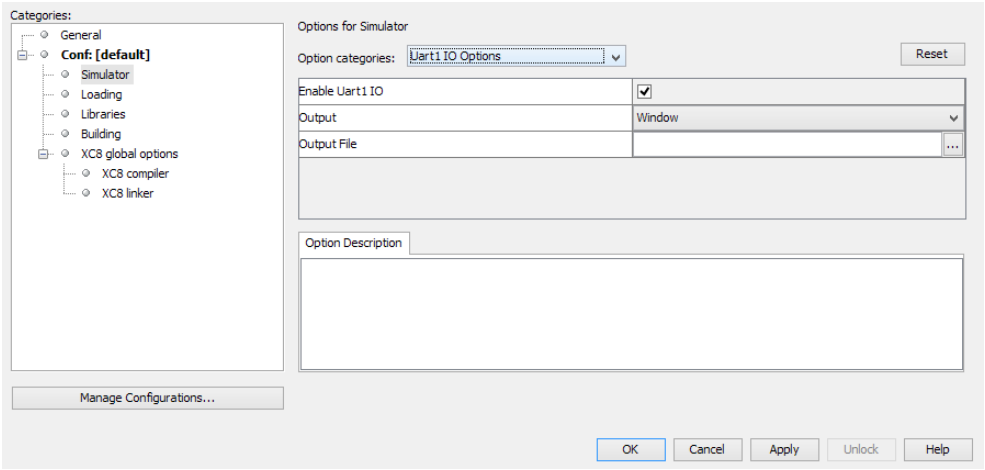


Figure 1 – MPLAB simulator configuration

(b) To test your setup (and your C skills), write a program displaying on the simulator the sum of the ten first even numbers (which should be 90).

The simulator results from the `printf` in your code are shown in the output window (at the bottom of the screen). Don't forget to call the `configure_uart()` function at the beginning of your code!.

[1] http://sin.begincoding.net/documentation/C/picebs2.h and http://sin.begincoding.net/documentation/C/picebs2.c

Question 4 – *Rotating led*

Write the code required to lit every single led after another on the first 4 bits on the `PORTH` of the simulator. You can display the status of each pin using *Window* → *Simulator* → *IOPin* and then select each of the pin you want to simulate. Even if the pin status is simulated, do not forget to set them as output pins first !

[Optional] Once a turn is over, bounce the led back in the other direction.

Question 5 – *Measuring performance*

As said in course, using a compiler does not change the underlying processor architecture.

(a) Write a program that divides two floating point numbers. Measure then time required to execute this code using the *stopwatch* (in MPLAB, under *Window* → *Debugging* → *Stopwatch*).

(b) With this information, how many divisions per second can you roughly perform?

. . . . . .

. . . . . .

. . . . . .