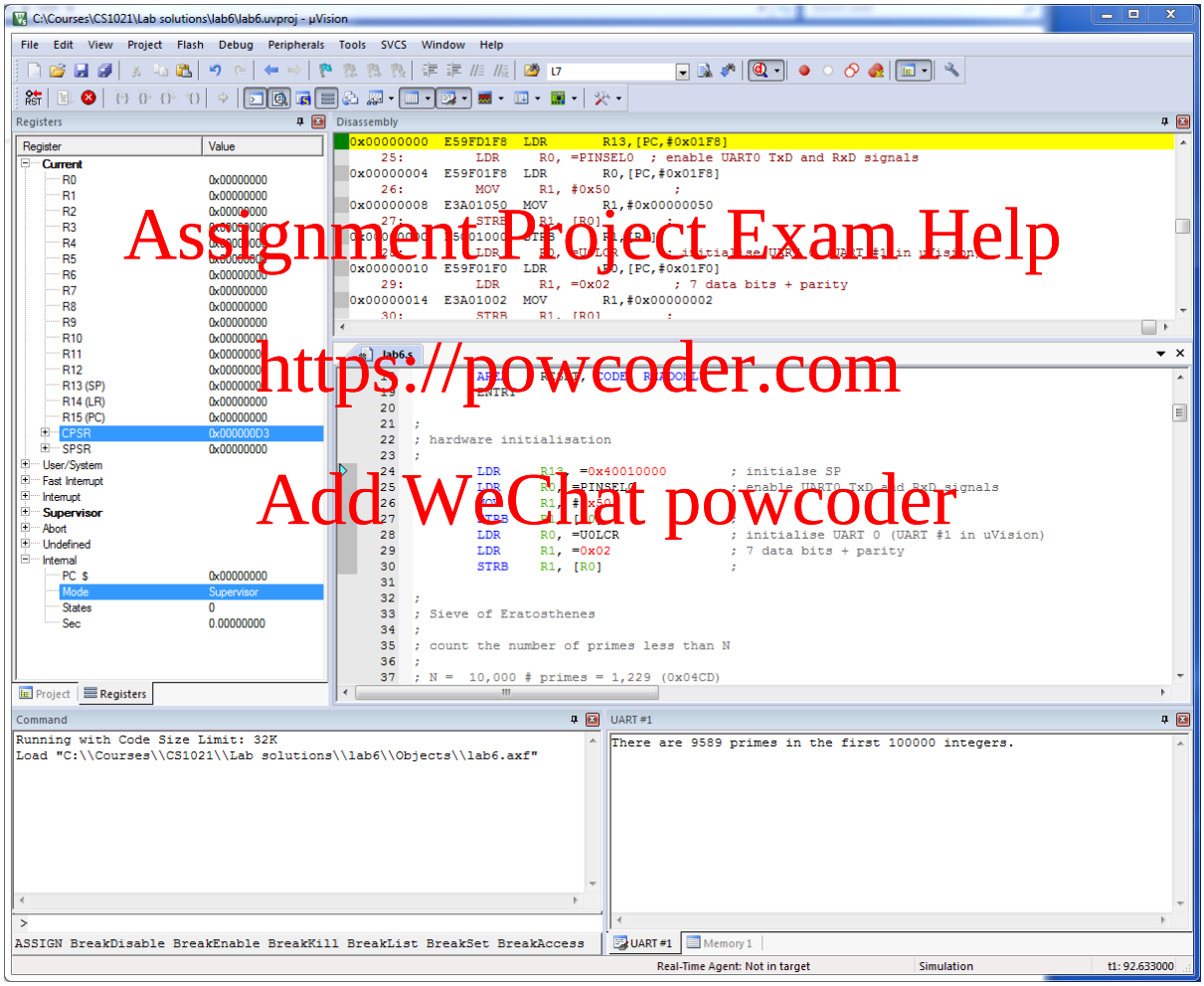
CS1021 Lab 6

Sieve of Eratosthenes

This is an individual Lab and will be marked out of 20.

Objective

The objective of lab6 is to use the "Sieve of Eratosthenes" to calculate the number of prime numbers in the first N integers. A sample screenshot from the program is shown below. It reports that there are 9,589 prime numbers in the first 100,000 integers.



Test your program with N = 10, N = 100, N = 1000, N = 10,000 and N = 100,000.

Algorithm

An **array of bits** (sieve) is used to represent the numbers 0 to N. if N = 100,000 then 100,000 bits are needed (12.5KB RAM). If a byte is used to represent each number, 100KB bytes of RAM is needed which is more than the 64KB of RAM available on the simulated hardware. All bits in the sieve are initialised to one. The pseudo code for the algorithm is shown below:

```
//
// create an array sieve[N] of N bits all initialised to 1
//
cnt = 0
for (int p = 2; p <= N; p++) {
    if (sieve[p]) {
        cnt++;
        for (int k = p*2; k <= N; k += p) {       // clear multiples of p
            sieve[k] = 0;
        }
    }
}
```


The algorithm tests each bit (p) in the sieve starting at bit 2. If sieve[p] is 1, it means that p is prime. The algorithm then clears the bits in the sieve corresponding to all multiples of p. For example if p = 3 and sieve[3] = 1, it indicates that 3 is prime. All multiples of 3 in the sieve are then cleared sieve[6] = 0, sieve[9] = 0, sieve[12] = 0 and so on. Eventually the sieve only contains ones which correspond to the prime numbers. The algorithm also counts the number of primes as they are found. The program must output the count of primes to the console as a decimal number using a subroutine PUTU, which has to be written.

PUTU Suggested Approach

One approach to output an integer as a decimal number is to divide the integer by decreasing powers of 10, using the quotient as the next ASCII digit and the remainder as the next integer to be divided by the next lower power of 10.

For example, given the initial value 4128

- dividing 4128 by 1000 will give a quotient of 4 and a remainder of 128. Therefore, 0x04 + 0x30 = 0x34 (character `4') will be the first ASCII character displayed.
- dividing 128 by 100 will give a quotient of 1 and a remainder of 28. Therefore, 0x01 + 0x30 = 0x31 (character `1') will be the second ASCII character displayed.
- and so on.

The initial power of 10 must be determined and the algorithm requires a division subroutine UDIV. Convert the divide algorithm developed in lab4 into a subroutine UDIV.

Pseudo-code for PUTU is shown below:

```
//
// PUTU
//
PUTU(int Q) {
    D = 1,000,000,000
    while (D > Q)              ; determine initial value for D
        D = D / 10;
    while (D > 0) {            ;
        R = Q % D;             ; modulus (remainder on integer division)
        Q = Q / D;             ; quotient
        PUT(char(Q));          ; output next digit
        Q = R;                 ;
        D = D / 10;            ;
    }
}
```

Extra Mile

Modifications to the approach described above that you might consider:

- Modify your code such that the bits in sieve represent odd integers ONLY (this will double the number of integers that can be stored in the sieve). See if you can determine the number of primes in the first 1,000,000 integers using the 64K of available RAM.

- Implement a more elegant, recursive version of PUTU, the pseudo code of which is shown below


```
//
// PUTU (recursive version)
//
PUTU(int Q) {
    R = Q % 10;
    Q = Q / 10;
    if (Q)
      PUTU(Q);             // call recursively to reverse characters
    PUT(char(R));          // output next digit
}
```

Submission

This lab will count towards your final CS1021 coursework mark. Submit your solution via Blackboard no later than **9am on Fri 30-Nov-2018**. You should submit lab6.s and one .pdf file which gives a brief summary of your work and a clear statement as to whether your code works or not. The .pdf file must include evidence that your program works (eg. use the Windows Snipping tool to get screen shots of the Keil IDE which show that your program generates the correct results).