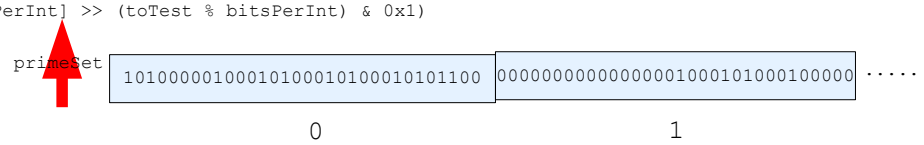
ARMAssemblyInt:BigEx:A

Primes.c

```
#include <stdio.h>

#define bitsPerInt 32U

static int checkNum(unsigned, unsigned[]);
void setBit(unsigned, unsigned[]);

int main() {
   const int numsToTest = 320;
   unsigned primeSet[10];          // Space for 320 bits
   int totalPrimes = 0, toTest, idx;

   for (idx = 0; idx < 10; idx++)
      primeSet[idx] = 0;

   for (toTest = 2; toTest < numsToTest; toTest++)
      if (checkNum(toTest, primeSet))
         totalPrimes++;

   printf("Found %d primes:", totalPrimes);
   for (toTest = 0; toTest < numsToTest; toTest++)
      if (primeSet[toTest/bitsPerInt] >> (toTest % bitsPerInt) & 0x1)
         printf(" %d", toTest);

   printf("\n");

   return 0;
}

static int checkNum(unsigned toTest, unsigned primeSet[]) {
   int divisor;

   for (divisor = 2; divisor < toTest && toTest % divisor != 0; divisor++)
      ;

   if (divisor == toTest)
      setBit(toTest, primeSet);

   return divisor == toTest;
}

void setBit(unsigned prime, unsigned primeSet[]) {
   primeSet[prime/bitsPerInt]
    = primeSet[prime/bitsPerInt] | (1 << prime % bitsPerInt);
}
```

```
00101010001000001000101000100000 primeSet[toTest / bitsPerInt]
00000000000000000000000101010001 primeSet[toTest / bitsPerInt] >> (toTest % bitsPerInt)
00000000000000000000000000000001 primeSet[primes / bitsPerInt] >> (toTest % bitsPerInt) & 1
```




```
01101100  A
10100110  B
11101110  A | B
```

```
00000000000000001000101000100000 primeSet[primes / bitsPerInt]
00000000001000000000000000000000 1 << prime % bitsPerInt
-----------------------------------
00000000001000001000101000100000 primeSet[primes / bitsPerInt] | 1 << prime % bitsPerInt
```