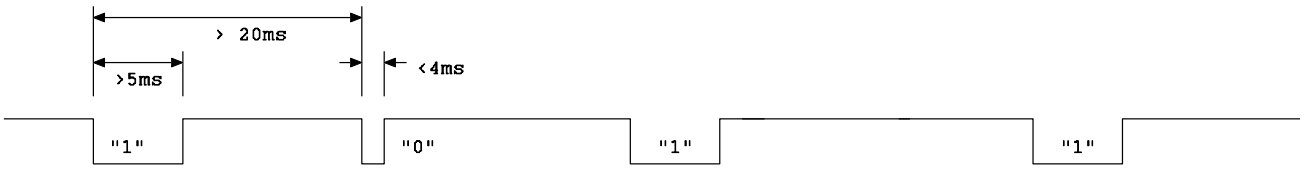
NAME: _____ STUDENT #: _____

EECE 259: Introduction to Microcomputers **Lecture Quiz** **Mar 23, 2011**

A one-finger keyboard! You can enter 1 byte of data into a computer serially, one bit at a time, using just a single switch. By holding the switch closed for a long time (say, >5ms), you can enter a binary '1'. However, by holding it closed for a short time (say, <4ms), you can enter a binary '0'. To be sure you won't miss any data, you know the spacing between two bits will be at least 20ms apart. Design a flowchart (or C program) and write an assembly language program that inputs exactly ONE byte of data from ONE switch and displays it on the 8 green LEDs. Assume the MSB arrives first. You can detect the arrival of each bit using polling. Use the built-in 50MHz counter to get nearly precise delays.




```
.include "ubc-de1media-macros.s"
.global _start
.text
_start:
```

NAME: _____ STUDENT #: _____

EECE 259: Introduction to Microcomputers **Lecture Quiz** **Mar 23, 2011**

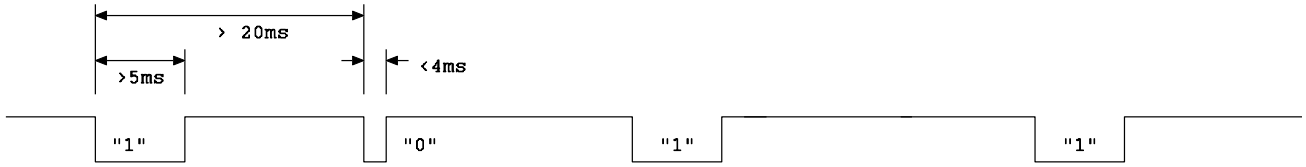
Design a flowchart (or C program) and write an assembly language program to compute the first N prime numbers. Store these numbers in a list in memory, starting at label PRIMES.

```
.include "ubc-de1media-macros.s"
.global _start
.equ        N, 50
.data
PRIMES:
.skip 4*N

.text
_start:
```

NAME: _____ STUDENT #: _____

EECE 259: Introduction to Microcomputers **Lecture Quiz** **Mar 23, 2011**

A one-finger keyboard! You can enter 1 byte of data into a computer serially, one bit at a time, using just a single switch, e.g. KEY3. By holding the switch closed for a long time (say, >5ms), you can enter a binary '1'. However, by holding it closed for a short time (say, <4ms), you can enter a binary '0'. To be sure you won't miss any data, you know the spacing between two bits will be at least 20ms apart. Design a flowchart (or C program) and write an assembly language program that inputs exactly ONE byte of data from ONE switch and displays it on the 8 green LEDs. Assume the MSB arrives first. You can detect the arrival of each bit using polling. Use the built-in 50MHz counter to get nearly precise delays.




```
#include "259macros.h"

int getBit()
{
      int start, cycles;

      // wait for KEY3 to be pressed
      while( *pKEY & 8 )
            ; /* do nothing in loop */

      // wait for KEY3 to be released
      // & measure time while it is pressed
      start = *pCOUNTER;

      while( ! *pKEY & 8 )
            ; /* do nothing in loop */


      cycles = *pCOUNTER - start;

      // 225000 = 4.5ms
      if( cycles < 225000 )
            return 0;

      return 1;
}
```

```
.include "ubc-de1media-macros.s"
.global _start
.text

getbit:

                  ldwio   r3, KEY(r23)
                  andi    r3, r3, 0x8
                  bne     r3, r0, getbit

                  ldwio   r8, COUNTER(r23)

wait:             ldwio   r3, KEY(r23)
                  andi    r3, r3, 0x8
                  beq     r3, r0, wait

                  ldwio   r9, COUNTER(r23)
                  sub     r9, r9, r8

                  movia   r3, 225000
                  bge     r9, r3, return1
return0:          movi    r2, 0
                  ret
return1:          movi    r2, 1
                  ret
```

```
int main( int argc, char *argv[] )
{
      int i, b, v=0;

      for( i=0; i<8; i++ )
      {
            b = getBit();
            v = (v<<1) | b;
      }

      *pLEDG = v;
      return v;
}
```

```
_start:           movia   r23, IOBASE
                  movi    r16, 8
                  movi    r17, 0

nextbit:          call    getbit
                  slli    r17, r17, 1
                  ori     r17, r17, r2
                  subi    r16, r16, 1
                  bgt     r16, r0, nextbit

done:             stwio   r17, LEDG(r23)
stop:             br      stop
```

NAME: _____ STUDENT #: _____

EECE 259: Introduction to Microcomputers **Lecture Quiz** **Mar 23, 2011**

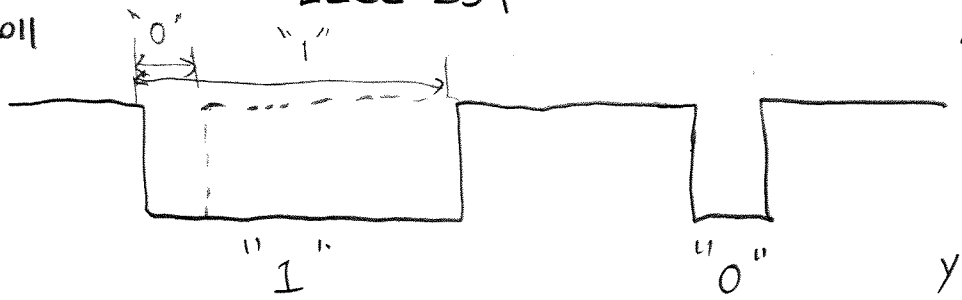
Design a flowchart (or C program) and write an assembly language program to compute the first N prime numbers. Store these numbers in a list in memory, starting at label PRIMES.

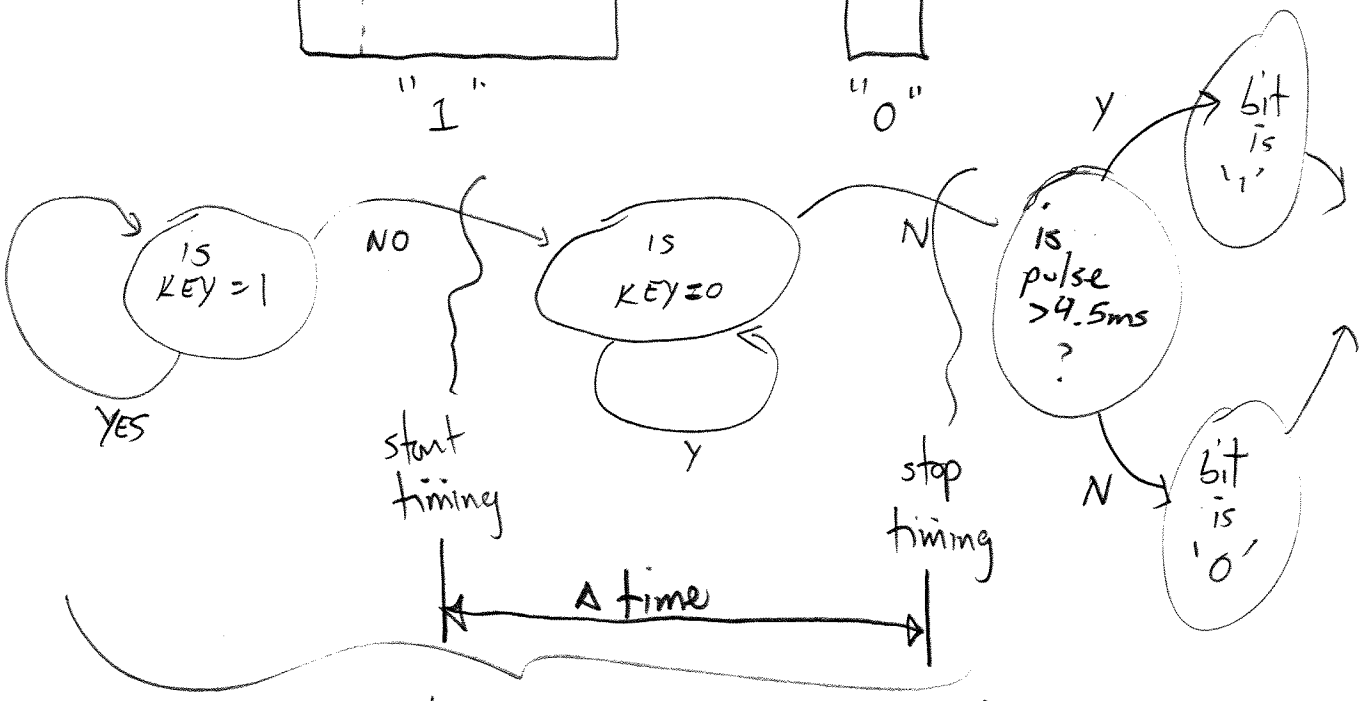
```
#include "259macros.h"

#define N 50
int primes[N];

int evenlyDiv( int x, int y )
{
      if( x/y*y == x )
            return 1;

int evenlyDivides( int x, int y )
{
      if( x/y*y == x )
            return 1;
      return 0;
}


int isPrime( int i, int num_primes )
{
      int k;
      if( i  < 2 ) return 0;
      if( i == 2 ) return 1;

      for( k=0; k < num_primes; k++ )
      {
            if( evenlyDiv(i,primes[k]) )
               return 0; // not prime!
      }

      return 1; // not divisible, so prime!
}

int main( int argc, char *argv[] )
{
      int i, num_primes;
      primes[0] = 2;
      num_primes = 1;
      i = 3;

      while( num_primes < N )
      {
            if( isPrime( i, num_primes ) )
               primes[num_primes++] = i;
            i+=2;
      }

}
```

```
.include "ubc-de1media-macros.s"
.global _start
.equ       N, 50
.data
PRIMES:
.skip 4*N

.text
_start:

evenlyDiv:  div     r2, r4, r6
            mul     r2, r2, r6
            bne     r2, r4, return0
return1:    movi    r2, 1
            ret
return0:    movi    r2, 0
            ret


isPrime:
            movi    r8, 2
            blt     r4, r8, return0
            beq     r4, r8, return1

            movia   r8, PRIMES
checkDiv:   ldw     r6, 0(r8)
            mov     r9, r31 /* save */
            call    evenlyDiv
            mov     r31, r9 /* restore */
            bne     r2, r0, return0
            addi    r8, r8, 4
            bltu    r8, r5, checkDiv
            br      return1


_start:     movi    r4, 2
            movia   r5, PRIMES

            stw     r4, 0(r5)
            addi    r5, r5, 4
            movi    r4, 3

            movia   r16, PRIMES+4*N
chkPrime:   call    isPrime
            beq     r2, r0, next
            stw     r4, 0(r5)
            addi    r5, r5, 4
next:       addi    r4, r4, 2
            bltu    r5, r16, chkPrime

stop:       br      stop
```

HINTS

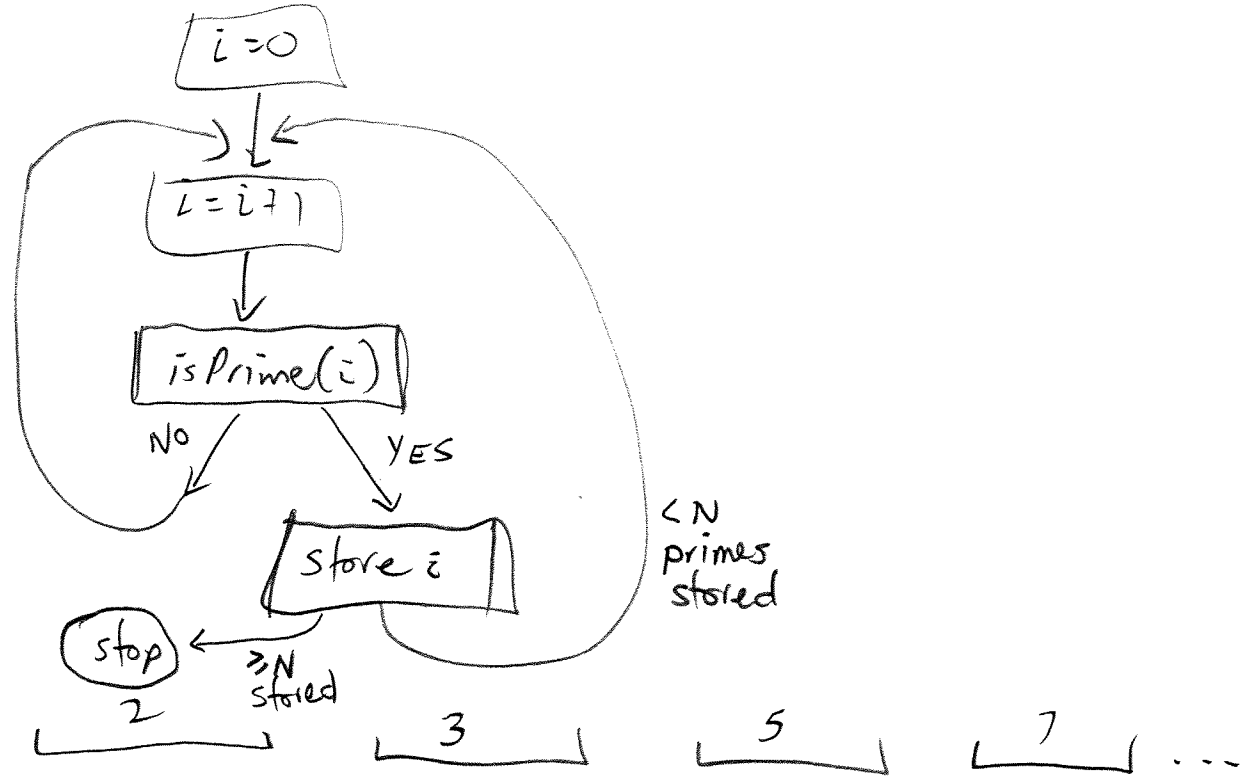
③



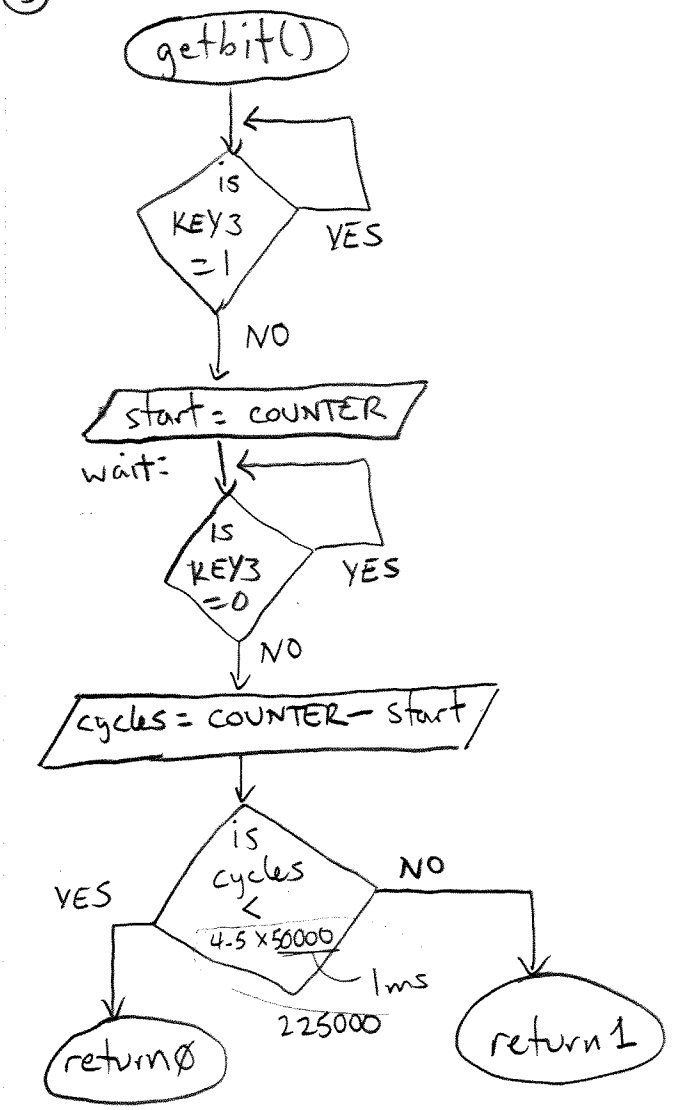


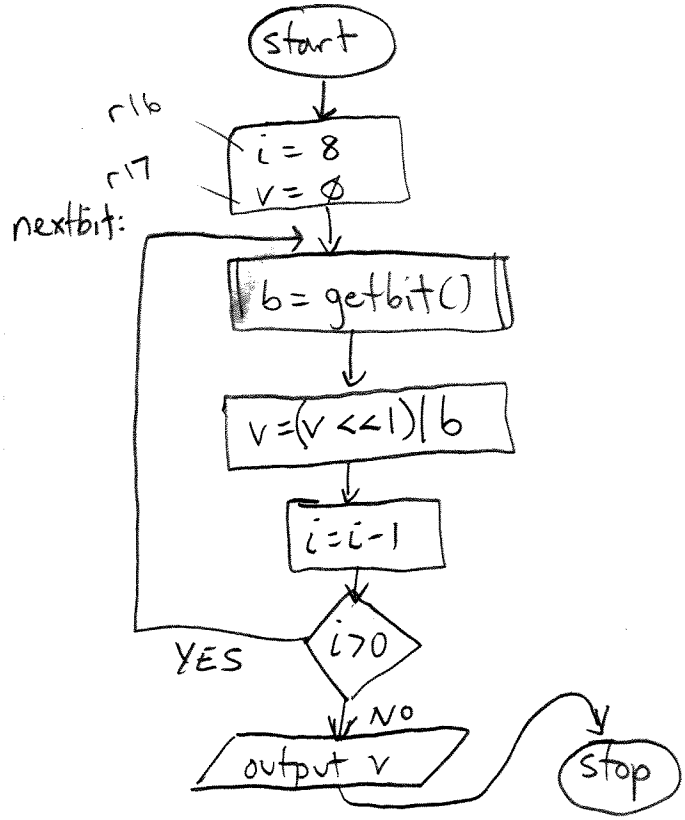
repeat 8x to form a byte + place on green LEDs.

④

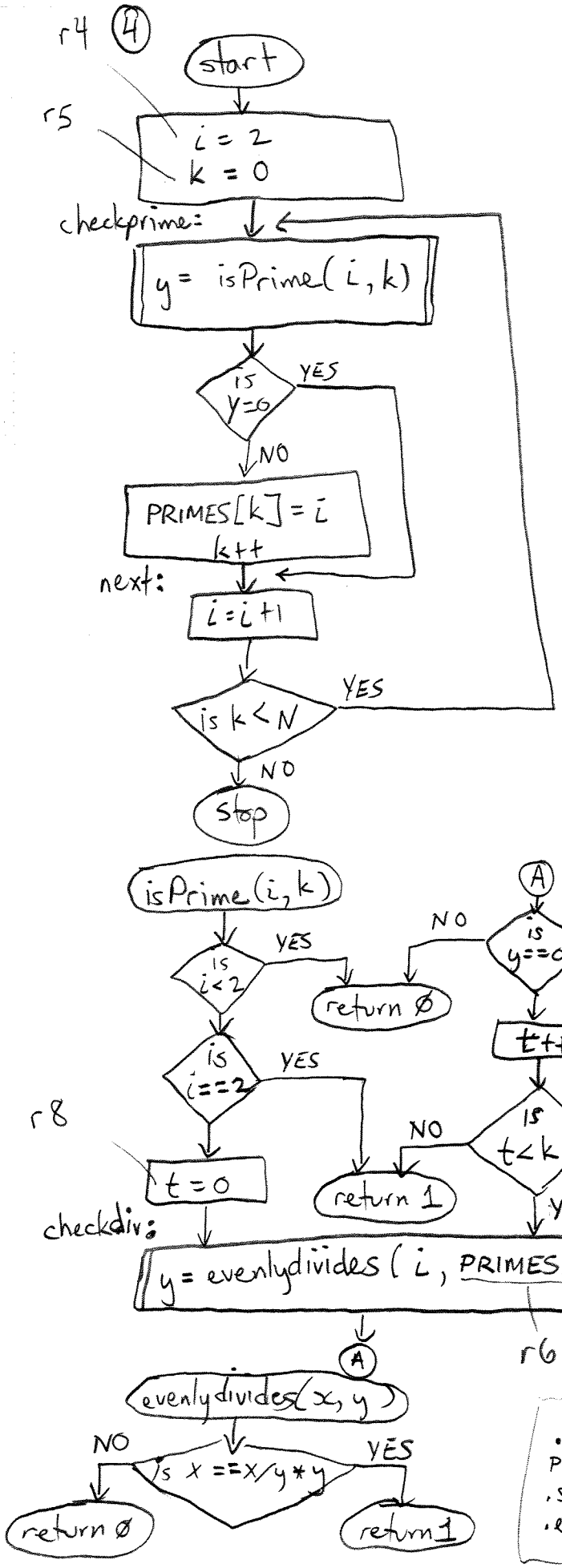


③






```
.text
.global _start

getbit: ldwio  r3, KEY(r23)
        andi   r3, r3, 0x8
        bne    r3, r0, getbit

        ldwio  r8, COUNTER(r23)

wait:   ldwio  r3, KEY(r23)
        andi   r3, r3, 0x8
        beq    r3, r0, wait

        ldwio  r9, COUNTER(r23)
        sub    r9, r9, r8

        movia  r3, 225000
        blt    r9, r3, return0

return1: movi  r2, 1
         ret

return0: movi  r2, 0
         ret

_start: movi   r16, 8
        movi   r17, 0
        movia  r23, IOBASE
nextbit: call  getbit

        slli   r17, r17, 1
        or     r17, r17, r2

        subi   r16, r16, 1

        bgt    r16, r0, nextbit

done:   stwio  r17, LEDG(r23)

stop:   br     stop
```




```
.equ N, 10 /* first 10 primes */
.text
.global _start
_start:   movi  r4, 2
          movia r5, PRIMES
          movia r16, PRIMES + N*4

checkprime: call  isPrime
          beq   r2, r0, next

          stw   r4, 0(r5)
          addi  r5, r5, 4

next:     addi  r4, r4, 1
          bltu  r5, r16, checkprime

stop:     br stop
```

```
isPrime:  movi  r8, 2
          blt   r4, r8, return0
          beq   r4, r8, return1

          movia r8, PRIMES

checkdiv: ldw   r6, 0(r8)
          mov   r9, r31      } save + restore r31 w/o stack
          call  evenlydivides }
          mov   r31, r9

          bne   r2, r0, return0
          addi  r8, r8, 4
          bgeu  r8, r5, return1
          br    checkdiv
```

```
evenlydivides: div  r2, r4, r6
          mul   r2, r2, r6
          beq   r2, r4, return1

return0:  movi  r2, 0
          ret

return1:  movi  r2, 1
          ret
```

```
.data
PRIMES:
.skip N*4
.end
```