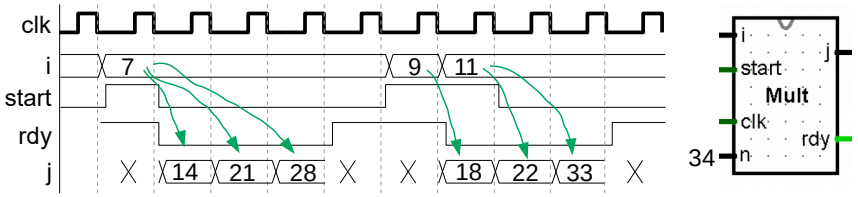
# CC4301 Arquitectura de Computadores – Examen – Otoño 2021 – Prof.: Luis Mateu

## Pregunta 1

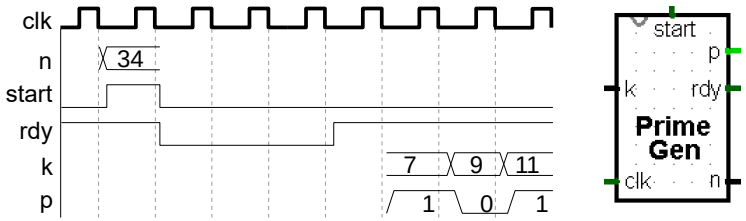
Implemente el circuito *Mult* de la figura. Debe entregar en la salida *j* los múltiplos del entero *i* que son menores o iguales a *n* (de 16 bits). El diagrama de tiempo muestra el funcionamiento de este circuito con la entrada n=34.




Cuando la entrada *start* se pone en 1, en el siguiente ciclo la salida *rdy* debe ir a 0 y deben aparecer los múltiplos de la entrada *i* por la salida *j*, de a uno por cada ciclo. Por ejemplo, los múltiplos de 7 (≤ 34) son 14, 21 y 28. Cuando el siguiente múltiplo es mayor que *n*, *rdy* debe ir a 1. Considere que la entrada *n* se mantiene constante durante la generación de los múltiplos.

## Pregunta 2

***Parte a.-*** El circuito *PrimeGen* es dado y genera los números primos impares ≤ *n* con el algoritmo de la criba de Eratóstenes. Para ello usa su circuito *Mult* de la pregunta 1. El siguiente diagrama muestra un ejemplo de uso:




Cuando *start* se pone en 1, comienza el cálculo de los primos impares hasta *n*. Cuando finaliza, *rdy* se coloca en 1. Entonces se puede consultar la primalidad de un entero ingresándolo por la entrada *k*. El circuito responde colocando *p* en 1 para indicar que *k* es primo o 0 en caso contrario. **No pierda tiempo durante el examen tratando de entender cómo funciona *PrimeGen*.**

Considere *PrimeGen* como un dispositivo de E/S. Implemente una interfaz para LRV32IM (incluido en los archivos adjuntos) que haga que al escribir *N* de 32 bits en el puerto con dirección 0x80000000, se active la línea *start* de *PrimeGen* y se suministre el valor *N* a la entrada *n*, haciendo que comience el cálculo de los primos. Necesitará truncar *N* a 16 bits con un separador por ejemplo.

***Parte b.-*** Programe la función *void start(int n)*, de modo que escriba *n* en el puerto 0x80000000. El cuerpo de la función necesita una sola línea de código.

***Parte c.-*** Extienda la interfaz de *PrimeGen* de manera que al leer el puerto con dirección 0x80000000 se obtenga el estado de la salida *rdy* (1 o 0). Necesitará extender *rdy* a 32 bits y conectarlo con un tristate al bus de datos.

***Parte d.-*** Programe la función *void waitRdy( )*, que hace un ciclo de busy-waiting leyendo el puerto para *rdy*, y retornando solo cuando *rdy* sea 1 (son 2 líneas).

***Parte e.-*** Extienda la interfaz de *PrimeGen* con 65536 puertos de lectura de 32 bits c/u, en el rango de direcciones [0x80040000,0x8007fffc], de tal manera que al leer el puerto con dirección 0x80040000+*K**4, se suministre *K* a la entrada *k* de PrimeGen y se obtenga su primalidad por la salida *p*. Necesitará extender *p* a 32 bits y conectarlo con un tristate al bus de datos.

***Parte f.-*** Programe la función *int isPrime(int k)* que entrega verdadero si *k* es primo *usando la interfaz de la parte e* (es una sola línea, calcular si *k* es primo por software no da puntaje). Si programa las funciones pedidas en el archivo *primegen.c*, podrá probar su implementación. Compile con *make primegen.ram* y ejecute en *LRV32IM*. Ingrese *n=34* y se mostrarán los primos ≤ 34 si respondió correctamente las preguntas 1 y 2.

## Pregunta 3

| *ciclo* | *fetch* | *decode* | *execute* |
|---|---|---|---|
| 0 | bge | lw | andi |
| 1 | | | |
| 2 | | | |
| ... | | | |
| 7 | | | |

***Parte i.-*** En los archivos adjuntos se incluye la ram *p3i.ram*, su "dump" en *p3i.dump* y el procesador *lrv32im-pipe* visto en cátedra. Cargue la ram en *lrv32im-pipe*, active el *breakpoint* y ejecute con *control-K* hasta que se detenga. Detenga el reloj y desactive el breakpoint. Luego ejecute ciclo a ciclo (con *control-T*) hasta la instrucción *ret* y complete la tabla de arriba, hasta el ciclo 7. En cada celda de la tabla escriba el nombre de la instrucción que se ejecuta en la etapa y ciclo correspondiente a esa celda.

***Parte ii.-*** Considere el diseño de LRV32IM que usó en sus tareas. El registro de instrucción IR tiene cargada la instrucción *ADDI A3, S3, 168* (A3 es el registro de destino). Evalúe de manera independiente si cada una de las 5 transferencias entre registros de la derecha se puede realizar en *un solo ciclo del reloj*. Si la transferencia se puede realizar indique cuáles son las señales de control requeridas para llevarla a cabo. Si no se puede realizar, explique por qué.

a) T ← S3+168
b) A3 ← S3-168
c) AR ← S3
d) IR ← S3
e) A3 ← PC+168

***Parte iii.-*** La figura muestra un extracto del contenido de un *cache* de 4 KB ($2^{12}$ bytes) de 1 grado de asociatividad con 256 líneas de 16 bytes. El computador posee un bus de direcciones de 16 bits. Por ejemplo en la línea a4 (en hexadecimal) se almacena la línea de memoria que tiene como etiqueta 7a4 (es decir, la línea que va de la dirección 7a40 a la dirección 7a4f).

| *línea cache* | *etiqueta* | *contenido* |
|---|---|---|
| a4 | 7a4 | |
| 3b | f3b | |
| 95 | c95 | |

Un programa accede a las siguientes direcciones de memoria: 7**a4**8, f**3b**4, d**95**0, f**3b**0, c**95**0, 5**a4**0, c**95**8. Indique qué accesos a la memoria son aciertos en el *cache*, cuales son desaciertos y rehaga la figura mostrando las etiquetas finales del cache.