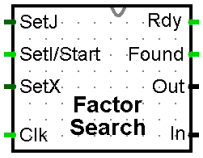
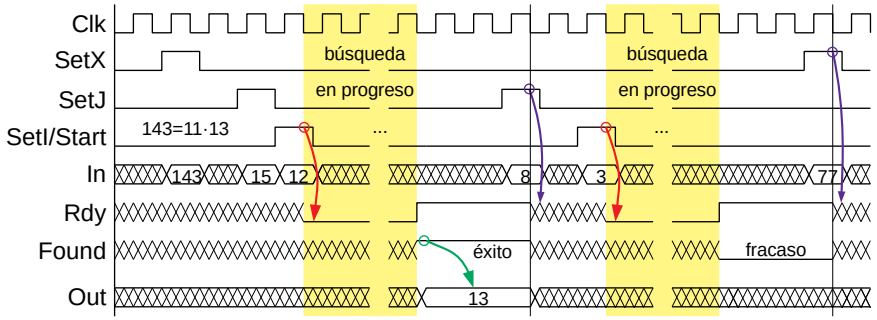
CC4301 Arquitectura de Computadores – Examen – Primavera 2021 – Prof.: Luis Mateu

Pregunta 1



Implemente el circuito *Factor Search* de la figura. Este circuito busca un factor de *x* en el intervalo [*i,j*] usando el algoritmo de la derecha. El parámetro *x* se especifica activando la entrada *SetX* y colocando su valor en la entrada *In* (de 32 bits). Para *j* se activa *SetJ* y se coloca su valor en *In*. Para *i* se activa *SetI/Start* colocando su valor en *In*, con lo que además se ordena el inicio de la búsqueda. Por lo tanto, el parámetro *i* debe ser el último en especificarse. La salida *Rdy* permanece en 0 mientras la búsqueda esté en progreso. Termina cuando *Rdy* pasa a 1. Un 1 en la salida *Found* indica éxito y el factor encontrado se indica por la salida *Out* (de 16 bits). Un 0 en *Found* indica que no se encontró ningún factor en el intervalo [*i,j*]. Las salidas deben permanecer constantes hasta que se especifique un nuevo parámetro. El circuito *lrv32im-factor.circ* de los archivos adjuntos incluye una plantilla para la solución en el módulo *Factor Search*. Además **explica la solución en palabras.** Ud. debe traducir la explicación a un circuito funcional. El módulo *Test Factor Search* le permitirá probar su circuito. El siguiente diagrama de tiempo muestra un ejemplo de uso en donde primero se busca un factor de 143 entre 12 y 15, encontrando el factor 13, y luego no se encuentra ningún factor entre 3 y 8.

```
int factorSearch(
    int x, int i, int j){
  for(int k=i;k<=j;k++){
    if (x % k == 0)
      return k;
  }
  return 0;
}
```



Pregunta 2

Parte a.- Considere el circuito *Factor Search* como un dispositivo de E/S. Implemente una interfaz para LRV32IM, incluido en el módulo *Piped Lrv32im* de *lrv32im-factor.circ*. Estas son las especificaciones. Escribir *j* en el puerto 0xfffe0000 define el parámetro *j*, escribir *x* en el puerto 0xfffe0004 define el parámetro *x* y escribir *i* en el puerto 0xfffe0008 define el parámetro *i* e inicia la búsqueda. Leer del puerto 0xfffe0008 obtiene el factor encontrado (salida *Out* de *Factor Search*) en los 16 bits menos significativos, el valor de *Rdy* en el bit 16, el valor de Found en el bit 17 y 0 en los bits 31 a 18. Si completa la interfaz de selección (módulo *Factor Selector*) y el resto de la interfaz en el módulo *Piped Lrv32im*, puede verificar su funcionamiento con *control-r*, *control-k*. El *display* le indicará las búsquedas de prueba y si algún resultado es incorrecto. En el circuito se comenta los breakpoints que son útiles para depurar su solución.

Parte b.- Programe la función *int factor(int x, int i, int j)* que busca un factor de *x* en el intervalo [*i,j*]. Ud. debe hacer la búsqueda usando el circuito *Factor Search* de la pregunta 1 y su interfaz de la parte a de la pregunta 2. Puede probar su función escribiendo su código en el archivo *factor.c*, compilando con *make factor.ram*, cargando *factor.ram* en la ROM del módulo Ram de *lrv32im-factor.circ* y ejecutando con *control-r control-k*. La salida debe ser idéntica a lo que muestra el display en la parte a. La función *main*, que invoca la función *factor*, está en el archivo *test-factor.c*. Note que las direcciones de las instrucciones que escriben o leen los puertos ya no son las de la parte a. Para depurar necesitará ejecutar *make factor.dump* y buscar las direcciones que ocupa la función *factor*.

Pregunta 3

a) PC ← T2-14
b) S4 ← T2
c) T2 ← S4-14
d) IR ← T2-14
e) S4 ← PC

Parte i.- Considere el diseño de LRV32IM que usó en la parte b de la tarea 5. El registro de instrucción IR tiene cargada la instrucción *ANDI S4, T2, -14* (S4 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 la señales de control requeridas para llevarla a cabo *en el mismo formato que usó en la parte b de la tarea 5*. Si no se puede realizar, explique por qué.

Parte ii.- La figura muestra un extracto del contenido actual de un *cache* de 64 KB (2^{16} bytes) de un grado de asociatividad con 4096 líneas de 16 bytes. El computador posee un bus de direcciones de 24 bits (es decir 6 dígitos hexadecimales). Por ejemplo en la línea 73a del caché (en hexadecimal) se almacena la línea de memoria que tiene como etiqueta 3573a (es decir, la línea que va de la dirección 3573a0 a la dirección 3573af).

línea cache	*etiqueta*	*contenido*
3d1	123**d1**	
73a	35**73a**	
f01	1a**f01**	

A continuación el programa accede a las siguientes direcciones de memoria: 1a**f01**4, 12**3d1**c, 2a**73a**8, 1a**f01**8, 35**73a**0, 40**f01**c, 12**3d1**8, 35**73a**c. Indique qué accesos a la memoria son aciertos en el *cache*, cuáles son desaciertos. Por ejemplo el primer acceso es un acierto. Además, rehaga la figura mostrando las etiquetas de las líneas que almacena el caché después del último acceso (35**73a**c).