Finales OdC (Ultimas 4 mesas desde 10 de febrero 2023)

Contents

Tips

Primera parte

1er mesa

- Si no hay suficientes unos preferir agrupar en L en Karnaugh
- Prestar mas atencion al marcar que casillas son unos, si no por ahi te salteas un uno y ni cuenta te das

2da mesa

- Marcar puntos primero, luego hacer lineas.
- Dibujar todas las lineas de compuertas primero, aumentando medio cuadrito de distancia con respecto a la anterior
- Calcular el tamaño de chip necesario en base al ancho de la palabra que se tiene. No es lo mismo 64k con un palabra de 8 bits que 64k con palabra de 32 bits
- Siempre añadir todas las columnas en una tabla de verdad

3er mesa

- Cuando hay que encender un led en base a una secuencia poner X en el caso general de la entrada e ignorarla en el minitermino

Segunda parte

1er mesa

- Es más fácil convertir a decimal desde hexa
- Para duplicar un float hay que aumentar en 1 el exponente
- Prestar atencion a cuando se incrementa una variable en una loop
- Pensar en mas formas en las que se pueden hacer las mismas cosas en codigo ASM
- CBNZ = CBN'

2da mesa

- La intruccion B es muy versatil si piden hacer bit flip o complemento a 2
- Para aumentar o disminuir el exponente de un float armar numero adecuado con movz o lsl y sumar o restar al float

3ra mesa

- En ascii, la “A” es 65 y “a” es 65+32=97
- Un EOR con -1 hace un bit flip
- Pensar en todas las cosas que tiene que hacer la instruccion que queres implementar en la ISA

FINAL 1ER MESA

PRIMER PARTE

1)

Diseñar un circuito combinacional que reciba como entrada dos numeros de 2 bits en formato binario natural y produzca como salida la suma de ambos numeros. Note que la salida tendra 3 bits. USAR NOR DE DOS ENTRADAS

2)

Diseñar y esquematizar un banco de memoria (Memoria + circuito de mapeo) para un procesador de 16 bits de bus de datos y 32 bits de bus de direcciones. Con 16 kbytes de ROM en las mas bajas y 16 kbytes de RAM en las mas altas. El sistema debe tener la mayor cant de posiciones espejo, las memorias deben ser del mayor tamaño posible pero limitadas con un bus de datos de 8 bits.

3)

Diseñar un circuito secuencial que detecte la siguiente secuencia 2,1,3 de manera consecutiva. La entrada es un binario natural de 2 bits. Al detectar la secuencia enciende un led (el circuito da salida 1) para apagar el led(salida 0) debe recibir un 0 una vez detectada la secuencia. Una vez apagado el led el circuito comienza de nuevo. Diagrama de estados, ecuaciones de transiciones y salidas. Diagramas en bloques del circuito completo.

SEGUNDA PARTE

1)

Poliglota float32 IEE754 Instruccion que sea float32 y a su vez tenga significado

2)

Duplicar y cambiar de signo un float 32 (Multiplicar por -2)

3)

V O F No es posible hacer un Loop infinito a una misma linea con un BR

4)

Dado el siguiente programa:

```
ORG 0x0
MOVZ X0, #0x100
MOVZ X1, #0x200
MOVZ X2, #0x20 -> 32 Decimal
MOVZ X3, #0
MOVZ X4, #0
LOOP:
        ADD X8,X3,X3          //X8 = X3*2
        ADD X8,X8,X8          //X8 = X3*4
        ADD X8,X8,X8          //X8 = X3*8
        ADD X8,X0,X8          //X8 = X3*8+#0x100
        LDUR X8, [X8, #0]     //X8 = Arr[X8]
        ADD X4, X4, X8        //X4 =X4 + X8
        ADDI X3,X3,#1         //X3++
        SUBS XZR, X2, X3      //0 = X2-X3 (Levanta flags)
        B.NE LOOP             //X2 != X3?
END:
        STUR X4, [X1,#0]      //0x200 = X4
        ret
```

a) Indicar cuantas instrucciones se ejecutan

b) Optimizar el programa para que haga lo mismo pero con menos instrucciones

5)

Para el programa anterior sin optimizar escribir en hexadecimal la sucesion de las primeras 32 direcciones a las que accede

6)

Modificar el diseño de la ISA para que contemple CBZ y CBNZ la nueva señal de control CB contempla que 0 es CBZ y 1 es CBNZ

FINAL SEGUNDA MESA

Parte 1:

Ej1)

Hacer un circuito combinacional que recibe naturales de 4 bits y devuelve como salida el resto de la división de dicho natural por 3, por ejemplo el resultado de 0111 = 01 (notar que la salida tiene 2 bits). Dibujar el circuito combinacional usando compuertas nor, el menor número posible de estas(AYUDA del ejercicio: expresarlo como la suma de maxiterminos)

Ej2)

Crear un banco de memoria que recibe 18 bit de address y sale 32 bits de datos (sin posiciones espejo dejando el espacio sobrante para espacio reservado para futuras ampliaciones), con un banco de memoria de 256kbytes de ROM (en las posiciones más bajas de memoria) y un banco de 512kbytes de RAM (en las pos más altas de memoria) pero restringiendo la salida de datos a 16 bits por cada chip.

b) Hacer el mapa de memoria

Ej3)

Hacer un circuito secuencial que siga la siguiente secuencia 1,2,3,4,5,6,7; 1,2,3,4,5,6,7; ... cuando la señal de entrada es 1 y la secuencia 7,6,5,4,3,2,1; 7,6,5,4,3,2,1; ... cuando la señal de entrada es 0.

a) Hacer el diagrama de estados

b) Dar las ecuaciones del combinacional de estados y del combinacional de salida

c) Dibujar el circuito con lógica combinacional y FF-D

Parte 2:

Ej1)

Dar una instruccion que si la damos vuelta(bit a bit) es otra instrucción válida

Ej2)

Dar el codigo en assembler que multiplique por 4 el siguiente número en formato float32
1011 1111 0010 0000 0000 0000 0000 0000

Ej 2)

Dar el codigo en assembler que divida por 4 el siguiente número en formato float32

X= = 0 10000101 10010010000000000000000 = 100.5

Ej3)

Que hace el siguiente código:

```
bl 1
addi x30, x30, #8
br x30
```

Ej4)

Pasar el siguiente código en C que calcula el MCD a assembler

```
do
   if a>b
   a = a-b
   else
   b = b-a
   fi
while (a!=b)
```

Ej5)

Ensamblar las siguientes instrucciones teniendo en cuenta que se ensamblan a partir de la dirección a) org.0x4000 y b)org.0xF000000

a)

```
loop:
        addi x0, x0, #2
        cbz x0, EXIT
        subi x0, x0, #2
        b loop
EXIT:
```

b)

```
loop:
        addi x0, x0, #2
        cbz x0, EXIT
        subi x0, x0, #2
        b loop
EXIT:
```

Ej6) Modificar el esquema de la ISA para agregar la instrucción “bl” y dar el estado de las señales de control de dicha instrucción

FINAL TERCERA MESA LPM

final 12/08/2022

primer parte

1.

circuito combinacional que divide un natural de 4 bits por 4, implementar con cualquier compuerta, cualquier cantidad de entradas ej: 1/4 = 0, 15/ 4= 3, 4/4=1

2.

procesador de 20 bits de direcciones y 16 bits de datos, implementar 512kbytes de ram en las posiciones mas bajas y 1Megabyte de rom en las mas altas. Generar la mayor cantidad de espejos posibles. Salida maxima de 8 bits por banco. Dibujar mapa de memoria (Aprox)

3.

circuito secuencial 2,1,3 y una entrada de 4 bits natural, se prende el led con la secuencia y se apaga con dos 7 consecutivos

a) Dibujar el diagrama de estados

b) Dar las ecuaciones del combinacional de estados y del combinacional de salida(Codif estados)

c) Dibujar el circuito con lógica combinacional y FF-D

segunda parte

1.

dar una instruccion que haciendole complemento a 2 es otra instruccion valida

2.

te dan un paquete de 8 char (64 bits) y tenes que pasar los char que estan en minuscula a mayuscula y viceversa Suponiendo que el paquete se encuentra en la direccion 0x0 en adelante, hasta 0x40.

3.

cuantas ejecuciones realiza el programa y optimizar

```
MOVZ x0, 0x1000
MOVZ x1, 0x100
loop:
        LDURB x8, [x0,#0]
        EOR x2, x2, x2 // X2 = XZR
        SUBI x2, x2, #1 // 0 - 1
        EOR x8, X8, x2 // EOR X8 y -1
        STURB x8, [x0, #0]
        ADDI x0, x0, #1
        SUBIS x1, x1, #1

        B.NE loop
        RET
```

(Creo que al programa le faltan instrucciones, lo recuerdo mas extenso)

4.

el programa del ejercicio 3 accedia a memoria con ldurb, en este ejercicio te pide rehacer el programa pero para acceder de a 64bits

5.

implementar BL y completar la tablita

Final 4ta mesa 10-02-2023

Parte 1

No tengo las consignas, pero era lo mismo de siempre

Parte 2

1)

Dar 2 pares de instrucciones que esten a distancia hamming 1 (googlear que es, en el final decia la definicion) y sean equivalentes. Dar binario y decodificacion.

2)

Escribir programa LEGv8 que compare dos float32 y salte a la label MENOR si $X0 < X1$ (asumiendo que los floats estan guardados en X0 y X1)

3)

Explicar que hace el siguiente codigo:

```
BL 1
MOVZ X0, #1, LSL #16
ADD X30, X30, X0
BR X30
```

Nota: No explicar tan linea a linea que a Wolo no le gusta, explicar que hace el codigo en general o explicar de ambas maneras

4)

Dado el siguiente codigo LEGv8:

```
MOVZ X0, #0X100
MOVZ X1, #0X200
MOVZ X2, #0X20
LOOP: LDUR X7, [X0, #0]
      CBZ X7, ZERO
      STUR X7, [X1, #0]
      ADDI X0, X0, #8
      ADDI X1, X1, #8
      B CONT
ZERO: ADDI X0, X0, #8
      CBNZ LOOP
END:  RET
```

a) Describir que hace

b) Determinar cantidad de instrucciones, siendo Z igual a la cantidad de ceros

c) Optimizar codigo

5)

Listar 32 instrucciones a las que accede el codigo del ejercicio anterior incluyendo fetchs (Se puede asumir que no hay ceros)

6)

Implementar MOVN (como MOVZ pero hace un bit flip, googlear bien que hace) en la ISA y rellenar tablita con las señales del ALU control