

Presentación para el Colóquio

1) ¿Cómo implementaría un circuito que se comporta igual que una fórmula de lógica proposicional?

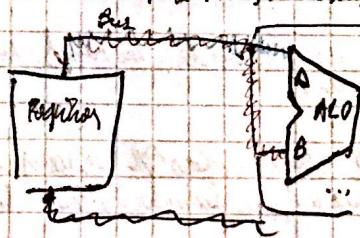
) Primero, intentaría reducir al máximo la expresión lógica con las técnicas propuestas vistas. De este modo, al obtener la expresión más reducida, puedo simplemente traducir de $X \cdot Y \Rightarrow X \rightarrow Y$

$$X + Y \Rightarrow X \rightarrow Y$$

2) ¿Qué función cumple la señal de reloj? ¿Qué circuito permite detectar el plazo vencido?

) La señal de reloj sirve para darutar un orden entre instrucciones. Por ejemplo, si se tiene que sumar el valor de R_0 al registro A de la ALU y el valor de R_1 al valor B de la ALU, se debe hacer por turnos. De otro modo, habría dos valores distintos en el bus, causando un cortocircuito.

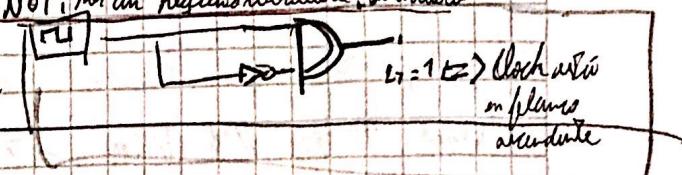
$$ADD R_0, R_1 \Rightarrow$$



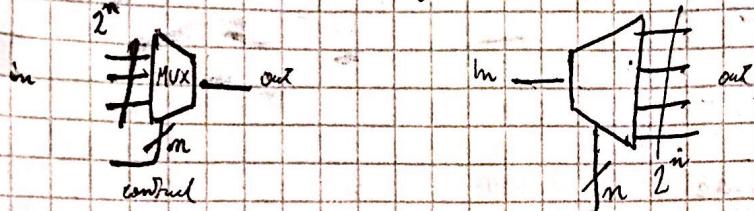
Respecto de ese plazo vencido, un Flip-Flop D podría servir. Para este propósito, queremos que al recibir el plazo vencido del clock, actualice su salida registrando el bit D (el dato guardado).

NOTA: P.ej., cuando se envía el clock, el FLIP-FLOP seta parar D y lo devuelve en la salida.

2) Para detectar el flanco ascendente dentro del "Flip-Flop D" se usa este pequeño circuito que se aprovecha de los tiempos de propagación. Al parar la señal NOT, un pequeño momento, el circuito dará output 1. Este pequeño momento corriente con el flanco ascendente del clock, y durará tanto que tocase el NOT en actualización y propagación.



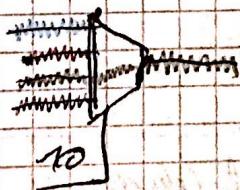
3) ¿Qué es un Multiplexor y un Demultiplexor?



Un multiplexor recibe 2^n posibles líneas de datos de entrada y n líneas de control.

Las líneas de control especifican cuál de las entradas es la que saldrá del Multiplexor.

Control	Línea Out	in
00	0	
01	1	
10	2	
11	3	



Un Demultiplexor hace exactamente lo contrario: Dada una entrada, las líneas de control determinan por cuál de las 2^n salidas saldrá.

Control	Línea Out
00	0
01	1
10	2
11	3



10 | 3 . . . 01] 153

a) ¿A qué le llamamos Estado de un procesador?

- .) Se le llama "Estado" a la situación actual en la que se encuentran los registros, los flags (en las unidades), el PC, la instrucción que se está ejecutando actualmente, las señales de control para ir a otro, pero sin perder el Estado actual. Para esto se guarda el Estado.

.) Memoria? Pila?

b) ¿Qué función cumple la señal load-microOp en microTsmall? ¿para qué es le llamamos al microPC?

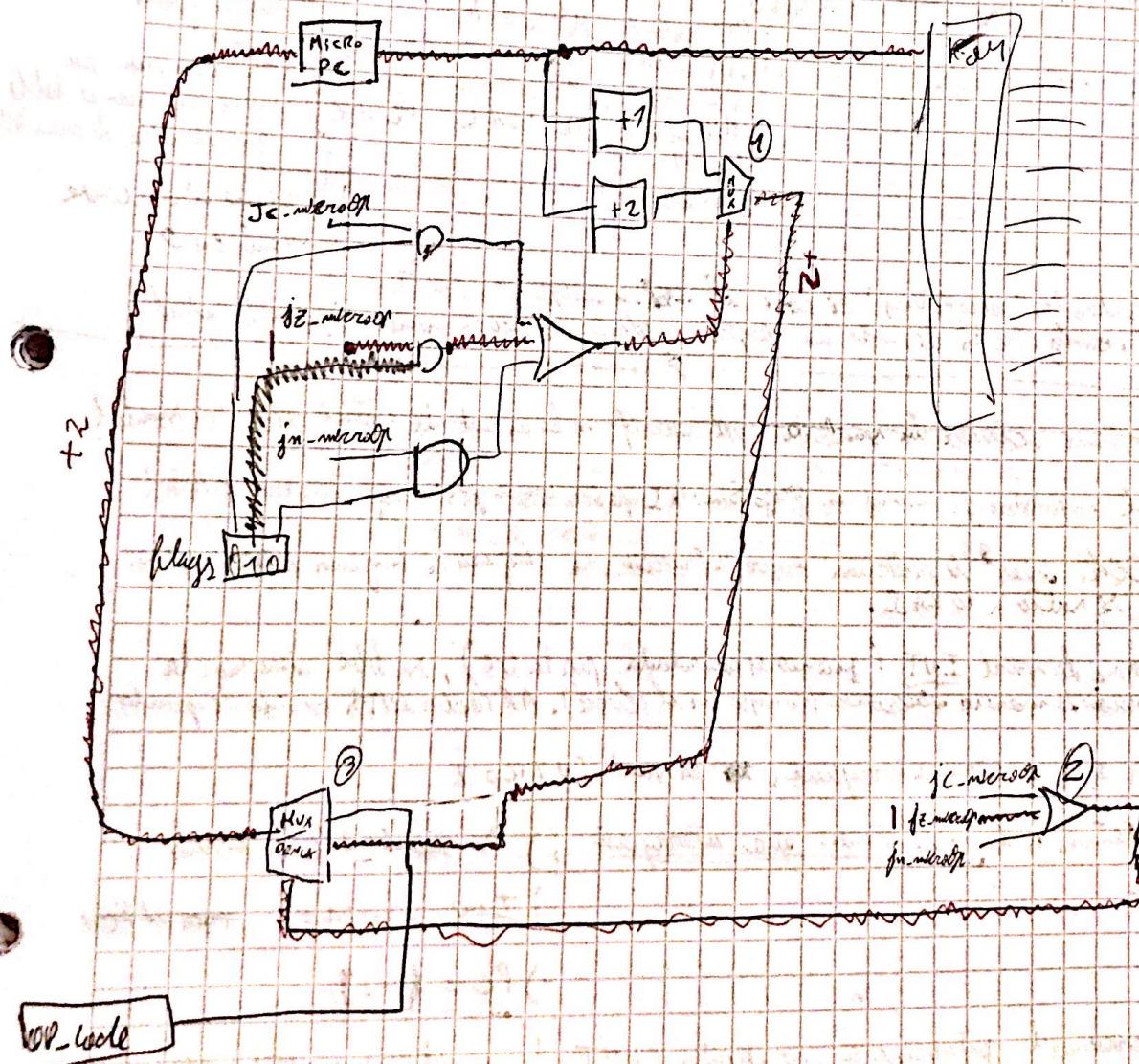
.) La señal load-microOp es la que permite que se pida el MicroPC con la dirección de memoria de la memoria de la operación que entra como opnd en OP-CODE.

.) El MicroPC es un contador, cuando cada ciclo de Clock aumenta en uno. Este registro permite saber qué micro-instrucción se está ejecutando en el momento ojo que señales va a "llamar" la UC.

NOTA

6) ¿Qué mecanismos tenemos en Orga 15 small para implementar saltos condicionales?

→ Dado que tenemos 3 saltos condicionales en la ISA, tenemos los flags para detectar si el salto se va a hacer o no.



En este caso, el MicroPC está indicando que se ejecuta la revisión 13, por ejemplo, se va a activar

OP-Code

• En Intel, cuando el MicroPC está midiendo que se activa la señal JZ , por ejemplo, se va a activar la señal JZ . Asumimos para este ejemplo que ~~Z=1~~ $Z=1$

• Luego, se dejará pasar en (1) a la señal que enviaría en 2 al microPC en vez de a 1. Además, en (3), se dejará pasar la señal en vez de a la del OP-Code. De este modo, el microPC NO hará la suma de valor +2

↳ Pero, por qué? → los microinstrucciones de los saltos condicionales están hechas del siguiente modo.

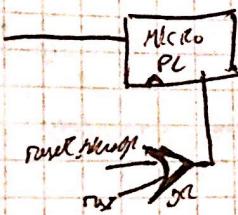
$JZ:$

+1 (load_MicroOp JZ -MicroOp
next_MicroOp
DE_out Imm PC-load) +2
NEXT-MicroOp

NOTA: • De este modo, el MicroPC cuando $Z=1$.

7) ¿Qué efecto tiene activar la señal reset-MicroOp en una T-mall?

1



- .) Si se activa reset-MicroOp, el MicroPC pasa a ver 0. De este modo, se pasa a la siguiente instrucción del PCs y que la instrucción 0 de microCode.apr era

000000:

PC_enOut	MM_enAddr	DE-load H	PC_inc
MM_inOut	MM_enAddr		
PC_enOut	MM_enAddr	DE-load L	PC_inc
MM_inOut	DE-load L		
[load MicroOp]			I
[next MicroOp]			J

birgen las tres instrucciones.

Toma tanto para leer una instrucción como el doble que las addrs de memoria.

carga el op_code
vuelve a 0.

- .) Nota, en todos los "micro Routines" se hace el MFT-MicroOp como un "empty check", de modo que siempre repite una instrucción para determinar la actual.

8) ¿Cuál es el mecanismo que tienen en Orga 154001 I para comenzar a efectuar una RTI?

) En el Fetch, antes de recibir una nueva instrucción, se checa si hay una interrupción. Si la hay, se salta a la RTI.

) Al receber la señal INT (que no es controlada por la UC), se debe almacenar la interrupción de manera atómica (sin importar el clock). Al recibir INTA, se deja de guardar.

) Luego, si $I=1$ y hubo una interrupción, la señal INTR=1.

) En el Fetch, si hay que atender una interrupción →.) Se guarda el PC en la R16

) $I=0$, INTA=1, ~~INTA=0~~

) $PC = \{R17\}$.

) Al terminar la interrupción, se efectúa IRET (salto del programador) y se vuelve al estado anterior a la interrupción.

9) ¿Cuál es la ventaja de tener operaciones de 3 registros en RISC-V?

→ Optimización del código; menos instrucciones para lo mismo.

→ Reducción del acceso a memoria; menos ciclos de clock para la memoria.

→ Más flexibilidad en las operaciones;

10) ¿Cuál suele ser el caso en otras arquitecturas?

→ Muchas arquitecturas RISC tienen operaciones de 3 registros.

NOTA: → lo que no, el compilador generalmente aplica "pseudo-instrucción" para dejar el mismo efecto con registros temporales.

		HOJA N°
		FECHA

11) En RISC-V ¿Qué significa que un registrador temporal?

.) Estos registros no tienen una capacidad de los demás, sino que por convención se los llama así:

.) Estos registros se utilizan para guardar valores intermedios en un cálculo.

.) Su contenido no se conserva entre llamadas a funciones y se utilizan según las necesidades del compilador por.

.) Por ejemplo, en el pseudocódigo se usa

ANDI \$r0, \$y, 1

BEQZ \$0, next-Iteración.

...

por. Al lo era,

.) lo que hace este código era chequear si \$y era mayor a \$r0, si lo era, dirigirse a next-Iteración.
.) Si el registro \$0 para guardar la información ya que no se intercalaba guardarla para una próxima iteración del ciclo.

9