FINALES

- Un procesador ARC ejecuta la instrucción assembler: srl %r10, 12, %r7.

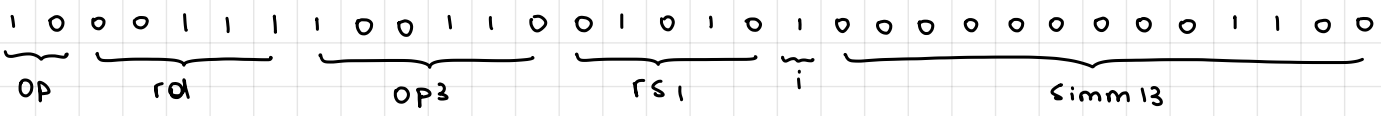
Se pide:

a) Escribir un microcódigo que implemente esta instrucción de assembler indicando en qué direcciones de la memoria de control estará almacenado.

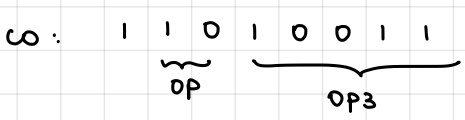
b) Para las siguientes líneas del microcódigo propuesto indicar los valores binarios (o su equivalente en decimal o hexadecimal) en los siguientes puntos de la microarquitectura:

- Entradas y salidas de la lógica de control de saltos
- Líneas de datos del bus A
- Todos los bits del registro de microinstrucciones
- CS Address MUX

Construyo el formato del registro %ir:



Decodifico: 1 1 0 1 0 0 1 1 0 0 0 = 1688_{10} → el código comienza en 1688.



1688: if [IR[13]] THEN GOTO 1690;

1690: R[tempØ] ← SIMM13 (R[ir]);

1691: R[rd] ← SRL (R[rs1], R[tempØ]); GOTO 2047;

2047: R[pc] ← INCPC (R[pc]); GOTO Ø;

Ø: R[ir] ← AND (R[pc], R[pc]); READ;

1: DECODE;

1690‖:

A = 1 0 0 1 0 1 → registro %ir ≡ r37

AMUX = 0 → uso el registro que guarda el campo A de la MIR y no rs1

B = X
BMUX = X
} no se usa ⇒ no importa qué info vengo

C = 1 0 0 0 0 1 → registro %tempØ ≡ r33

CMUX = 0 → uso el registro que guarda el campo C de la MIR y no rd

RD = 0
WR = 0
} no leo ni escribo la memoria principal

ALU = 1 0 1 1 → el código en ALU para la operación SIMM13

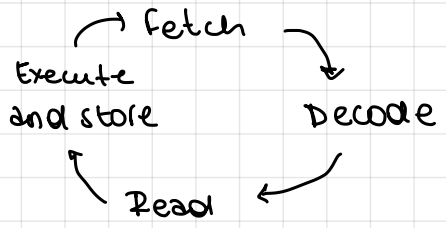
COND = 0 0 0 → la siguiente línea a ejecutar es la consecutiva

JAMP ADDR = X → no hay salto condicional ⇒ no importa qué info. vengo

18/02/2020

1) (a) Explicar qué es el ciclo de fetch y con qué finalidad se realiza.
(b) Explicar de qué manera se determina durante la ejecución de microcódigo la próxima microinstrucción a ser ejecutada.
(c) Dar varios ejemplos de instrucciones de assembler cuya implementación en microcódigo requiere pasar por la microinstrucción almacenada en la posición 2047. Dar varios ejemplos de instrucciones que no requieren pasar por esa línea de microcódigo. Justificar su respuesta.

a) El ciclo de fetch es el proceso por el cual un sistema ejecuta un programa. Es coordinado por la unidad de control y consta de los siguientes pasos:



El ciclo de fetch comienza con la búsqueda en la memoria principal (pues allí se almacena el programa) de la siguiente instrucción a ejecutar. Al comienzo, la ROM lee la microinstrucción 0 que lee y almacena en el registro %ir la instrucción a la que apunta el PC.

La siguiente microinstrucción que lee la ROM es la 1 que decodifica el registro %ir. Para esto, toma los bits 31, 30, 19-24 y coloca 3 fijos (1 en la posición 10 y 0 en las posiciones 0 y 1). De esta forma, el decode queda conformado por 11 bits, de los cuales 8 pertenecen a la información almacenada en el registro %ir y 3 son fijos.

La ROM lee el decode y ejecuta la microinstrucción que le corresponde. En este punto, el CS Address MUX, lee del Control Branch logic los bits 10 (en ese orden) que indican decodificación instantánea y es por eso que se realiza dicha acción

Una vez que la ROM leyó el decode, la MIR lee de la ROM la microinstrucción a ejecutar y almacena la información correspondiente para hacerlo Cada celda de la MIR corresponde a:

- A, B y C: registros que serán colocados en los Buses A, B y C respectivamente.
- MUX A, B y C: indica si el MUX del Bus que corresponda debe seleccionar (para colocar en el Bus) el registro del scratchpad o de la celda que le corresponde en la MIR.
  Si el MUX lee un 0 de la MIR, indicará al Decoder del Bus que seleccione el registro indicado por la celda correspondiente en la MIR.
  Si el MUX lee un 1 de la MIR, indicará al Decoder del Bus que seleccione el registro indicado por la instrucción inicialmente leída. El registro %ir almacena los celdas rd (para los formatos SETHI, Aritmético y de Memoria), rs1 y rs2 (para los formatos Aritmético y de Memoria). El registro rd será colocado en el Bus C cuando el MUX C lea 1 y los registros rs1 y rs2 serán colocados en los Buses A y B cuando los MUXES A y B lean 1 respectivamente.
- RD y WR: indica si la microinstrucción en ejecución requiere lectura y/o escritura en memoria según corresponda.