Marelli-Mana-Letini-Schenone

Tp3 - Microarquitectura

1. Leer la hoja de datos y responder:
    a. El tamaño de la memoria es de 256 byte
    Tiene al menos 3, ya que restando las instrucciones con operando y sus variantes, el opcode no cambia, los registros son susceptibles a cambiar las instrucciones.

    b. 8 bits.

    c. El IR se encuentra en el decode. A su vez se divide en 2 subregistros, IR-L para la parte baja, de 8 bits. Y IR-H para la parte alta, también de 8 bits. Sumando así en su totalidad una cantidad de 16 bits.

    d. Estaría en el Decode, tendría 16 bits, y como mínimo 8 bits. Debido a que estos se encuentran en la parte baja IR-L (8 bits) y IR-H (8 bits) para la parte alta
    Las microinstrucciones están en una memoria X, que forma parte del componente UC, contiene palabras de 32 bits y direcciones de 9 bits.

2)
    a. En el program counter, la función inc se refiere al incremento del puntero en la memoria, el cual apunta a direcciones de memoria, con cada incremento tendremos una memoria consecutiva.

    b. En nuestro ALU, opW es referido a considerar si deben escribirse las flags o no.

    c. Los saltos de Jump condicionales, toman distintas formas y sirven dentro de todo para poder redireccionar una cuenta o parte especifica del código que queremos que se ejecute y está escrito en otra parte. Existen varios tipos, por ejemplo el JN que ejecuta el jump si nuestro CMP (compare) fue negativo, sino el JZ que salta si el compare dio 0. Las instrucciones que el enunciado plantea, son conectadas a un cable que lleva hacia una compuerta AND, tenemos otro cable X, y quiere decir que cuando alguna de nuestras flags se active, la compuerta and tambien lo hara y ejecutara un “jump” hacia donde se indique.

    d. La señal DE_enOutlmm habilita la entrada al bus de un valor inmediato. Y el CU es el encargado de indicar a quien leer o escribir.

3.

```
start:
        SET R7, 0xFF
        SET R0, 0x01
        SET R1, 0x00
        SET R2, 0x10
        SET R3, 0x30
loop:
        CALL |R7|, shift
        SUB R2, R0
        CMP R1, R2
        JZ end
JMP loop
end:
JMP end
```

```
shift:
        PUSH |R7|, R2
        SHL R2, 1
        STR [R3], R2
        ADD R3, R0
        POP |R7|, R2
        RET |R7|
```

A) El programa es un bucle infinito que se llama a la instrucción shift. Dentro de ella, se realiza un shift lógico a la izquierda, es decir lo multiplica por dos en el registro R2. Almacena el valor resultante en una apuntada por R3. Luego se recupera el valor original de R2 desde la pila, y lo retorna a la llamada usando RET. El bucle principal no finaliza hasta que R1 sea igual a R2, por el CMP que realiza una resta explicita, el programa salta a "end:" e ingresa en un bucle infinito en ese lugar debido a la instrucción JMP end, donde salta y finalmente saltará a la etiqueta end una y otra vez hasta salir del bucle.
Si quisiéramos pensarlo en la práctica, podríamos ver que R2 en este caso empieza valiendo 16, el loop se ejecutará 16 veces en total, almacenando 16 valores de memoria.

B) Etiquetas:
- Start: |00|
- Shift: |16|
- Loop: |0a|
- End: |14|

C) Para que el programa ingrese al loop infinito por primera vez, se requieren 1515 ciclos de clock y para que llegue a la etiqueta de end, son 122 ciclos en el PC.

D) ADD tiene 5 micro instrucciones incluido el reset - JZ 4 Microinstrucciones con ambos reset comparados - JMP 2 Microinstrucciones incluido el reset

E) Describir detalladamente el funcionamiento de las instrucciones:
- PUSH: Guardo un dato en la pila para poder operar con él sin perderlo. Se le indica la dirección apuntada, luego guarda el dato en esa posición y resta uno, apuntando a la siguiente.
- POP: Recuperó un valor de la pila y se utiliza luego de un push, estrictamente. Con la idea de que “el último registro pusheado” será mi primer pop.
- CALL: Llama una función o una subrutina y la almacena en la pila. Antes de la subrutina actualiza el PC.

- RET: Recupera la dirección almacenada en la pila, transfiere el control a la dirección especificada y la carga en el PC. Siempre se hace después de una funcion osea despues de un CALL

4. En archivo

5. En archivo