Primer Recuperatorio Organización del Computador

FIUBA - 2024-2C

- Nombre y apellido:
- Padrón:
- Hojas adicionales entregadas:

Assembly

a) Implementar una función `get_invertebrados_por_region` en assembly que dada una lista simplemente enlazada, devuelva un array de `C` de la misma cantidad de elementos que la lista, con los índices ordenados de todos los animales que sean invertebrados y coincidan con la región pasada como parámetro. Una vez que se completaron los índices de los animales invertebrados, se deben completar los índices restantes con el valor `0xFF`. Un animal es invertebrado cuando el campo `invertebrado` es distinto de `0`. Tener en cuenta que:

- En los índices se debe considerar que el primer elemento de la lista tiene índice `0`.
- Recuerde que el array devuelto es nativo de C y no el tipo array que vimos en el taller de assembly.
- El array devuelto debe ser liberado por el código C que la llame. O sea, no es responsabilidad de la función liberar la memoria.

```
typedef struct animal {
    char* nombre;
    uint8_t invertebrado;
    char* region;
} animal_t;
```

La estructura de la lista es la siguiente:

```
typedef struct node {
    struct node* next;
    animal_t *animal;
} node_t;

typedef struct list {
    node_t* first;
    node_t* last;
    uint8_t size;
} list_t;
```

La función debe tener la siguiente firma:

```
uint8_t *get_invertebrados_por_region(list_t* list, char *region);
```

Ayuda: > Pueden utilizar la función `memset` de la biblioteca estándar de C.

Restricciones

- La lista puede estar vacía
- La lista no tendrá más de 255 elementos
- Se pueden usar las funciones del taller que considere útiles
- Se pueden usar funciones de la biblioteca estándar de C.
- La función debe ser implementada en assembly.

Memoria Cache

Hacer el seguimiento de escrituras y lecturas de la siguiente secuencia de direcciones, en una memoria cache con las siguientes características: Dirección de 5 bits y memoria direccionable a byte.

Características de la memoria:

- 4 sets
- 1 línea por set

- 4 bytes por bloque
- 1 byte por lectura/escritura

Suponga que la memoria cache está vacía al inicio y utiliza como política write-back / write-allocate. La política de desalojo es LRU.

op	address	t	s	b	hit/miss/dirty-miss
W	0x10				
W	0x08				
R	0x11				
R	0x18				
R	0x02				
R	0x14				
W	0x19				
W	0x1C				
R	0x03				
R	0x19				
R	0x13				
W	0x04				
R	0x07				

¿Cuál es el estado final de la memoria cache? ¿Cuántos accesos a memoria RAM se realizaron?

Microarquitectura

Se tiene una arquitectura con el siguiente esquema:

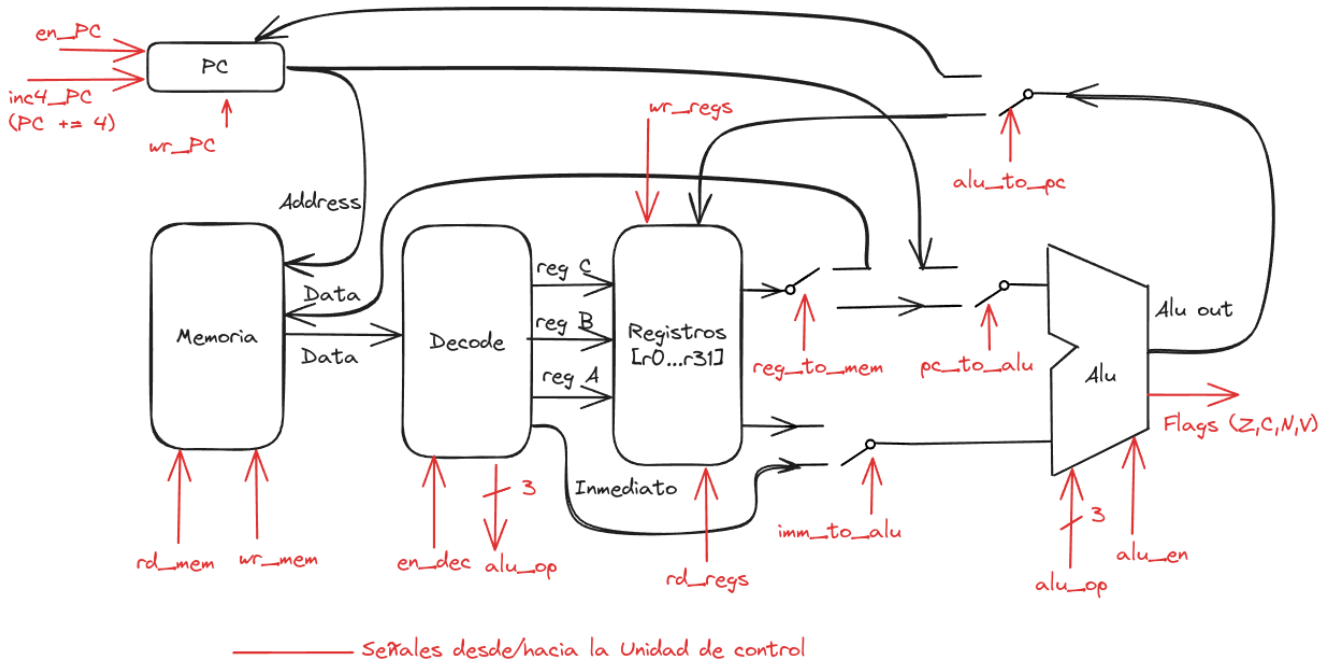

Figure 1: Esquema de la arquitectura

Con las siguientes características:

- Tamaño de palabra: 32 bits
- Tamaño de direcciones: 32 bits
- Tamaño de todas las instrucciones: 32 bits
- Direccionamiento a Byte
- 32 registros de propósito general

Para dicha microarquitectura, **se pide** modificar (sobre el esquema) la arquitectura y dar la secuencia de microinstrucciones para ejecutar una instrucción de movimiento de datos desde memoria a un registro, por ejemplo:

```
...
mov R2,[una_etiqueta]
...
```

Tener en cuenta que el ensamblador reemplaza el valor de `una_etiqueta` por la dirección de memoria correspondiente.

Suponer que la memoria tarda un clock en responder a una lectura, y de la misma forma, que todos los componentes ponen el dato correcto en su salida, un clock después de recibir la señal correspondiente.

Pueden suponer que las salidas de los bloques permanecen activas indefinidamente y se actualizan un clock después de modificarse alguna entrada.

las señales `reg_a`, `reg_b`, `reg_c`, `reg_d` son las señales de selección de registros. Es decir, si `reg_a=2` entonces el registro `R2` es el que se selecciona. Si `reg_a=0xFF` entonces no se selecciona ningún registro. `reg_a`, `reg_b` y `reg_c` se corresponden con operando1, operando2 y destino respectivamente. La unidad de decode se encarga de esto adecuadamente cuando lee una instrucción de forma correcta.

> *Utilizar el lenguaje de señales usado en el TP de microarquitectura con la* `orgaSmall`
>
> ***Explicar con sus palabras el funcionamiento de la secuencia de microinstrucciones.***

Paginación

Dar las traducciones a memoria física de las siguientes direcciones virtuales, según el esquema de paginación presentado a continuación:

Direcciones:

`0x0300A5F8; 0x03C0B042; 0x03C0BFFA;`

Estructuras de paginación:

Page Directory:

Entry	Page Table
8	0x00010
12	0x00020
15	0x00021

Page Tables:

0x00010:

Entry	Page Table
1	0x00022
2	0x00023
7	0x00024

0x00020:

Entry	Page Table
11	0x00026
12	0x00027

0x00021:

Entry	Page Table
10	0x00042
11	0x00043

Representación de la información y sistemas combinacionales

Sean A y B dos números de punto flotante de 32 bits en el formato IEEE 754.

- **Se pide:**
  - Diseñar el circuito que implementa la operación correspondiente a $A > B$ donde el resultado es un bit que vale `1` si se cumple la condición. Considerar a A y B como números de punto flotante positivos.

  Se pueden utilizar bloques implementados en el TP de sistemas combinacionales y/o secuenciales. No es necesario dibujar la estructura completa si hay muchos elementos repetidos, pero si aclarar cómo se conectan y la cantidad de elementos a utilizar.