Primer Parcial Organización del Computador

FIUBA - 2024-2C

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

Assembly

a) Implementar una función `stringify` en assembly que dada una lista simplemente enlazada, devuelva una representación en string de la misma. La representación debe empezar con el nodo `first` y terminar con el nodo `last`. Cada nodo debe imprimir el valor entero y estar separado del siguiente por el string `" -> "` (notar los espacios antes y después de las flechas). Cuando se llegue al final de la lista se debe imprimir el string `"NULL"`.

La estructura de la lista es la siguiente:

```
typedef struct node {
    struct node* next;
    int data;
} node_t;

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

La función debe tener la siguiente firma:

```
void stringify(FILE *fp, list_t* list);
```

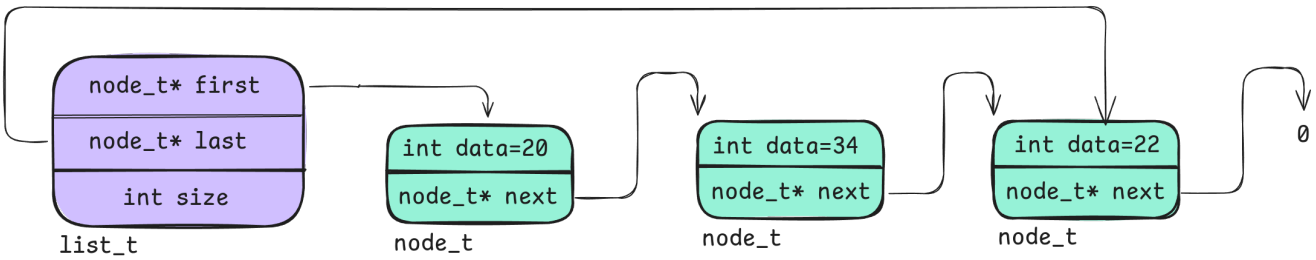
Donde `fp` es un puntero a un archivo abierto en modo escritura. La función debe escribir en el archivo la representación en string de la lista.

b) Escribir en C una rutina que llame a la función implementada en assembly, agregue 3 números a la lista y muestre por pantalla la representación en string de la lista. Para esto suponga que dispone de la siguientes funciones ya codificadas en C:

```
list_t* create_list();
void add_node(list_t* list, int data);
void destroy_list(list_t* list);
```

Ejemplo

Dada la lista:



La representación en string de la lista sería:

```
20 -> 34 -> 22 -> NULL
```

Restricciones

- La lista puede estar vacía, cuyo caso se debe imprimir `"NULL"`.
- La función debe ser implementada en assembly.
- Los números de la lista serán estrictamente enteros positivos.

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
- 2 líneas por set
- 2 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	0x18				
R	0x02				
R	0x11				
W	0x1C				
R	0x19				
W	0x19				
R	0x03				
W	0x04				
R	0x07				
R	0x13				
R	0x14				

¿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:

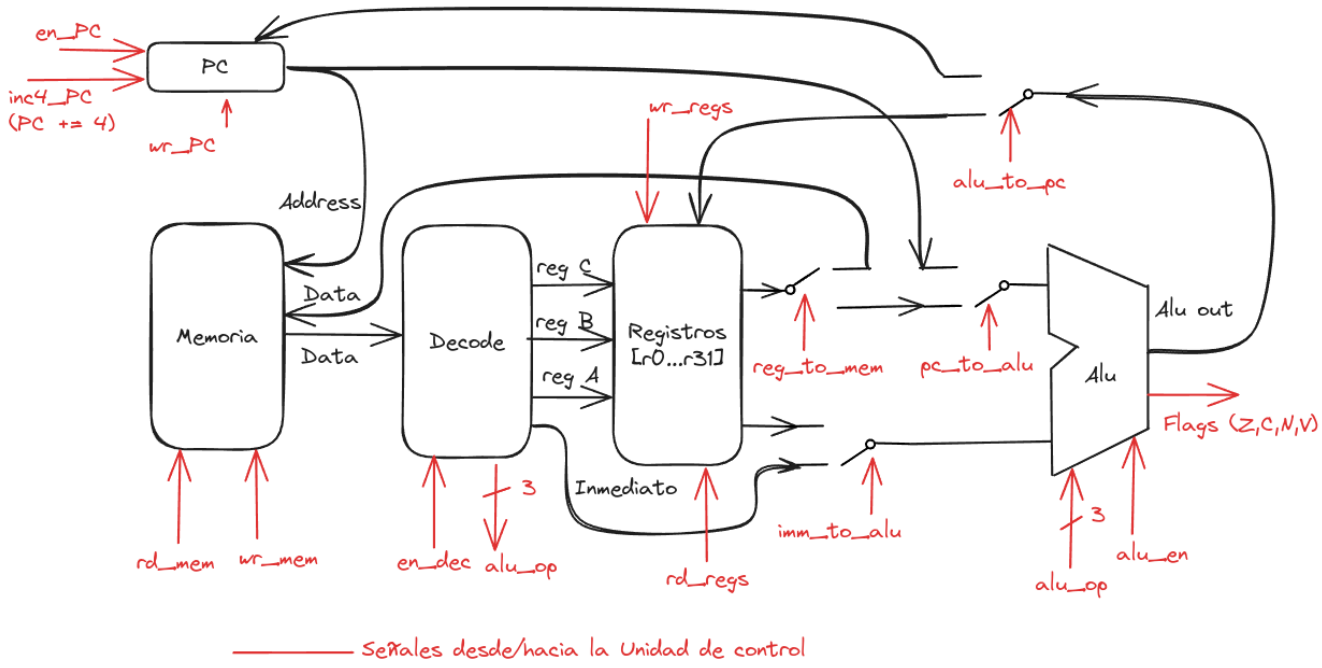


Figure 1: Esquema de la arquitectura

Con las siguientes características:

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

Para dicha microarquitectura, **se pide** dar la secuencia de microinstrucciones para ejecutar una instrucción de salto incondicional, por ejemplo:

```
...
jmp una_etiqueta  ; Esta instrucción hay que implementar
...

una_etiqueta:
...
```

Tener en cuenta que el ensamblador calcula el valor de `una_etiqueta` como un desplazamiento del `PC` actual, i.e. lo reemplaza por un número para sumar al `PC` de la posición actual.

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 por un clock o por siempre (actualizándose el valor con la señal de control correspondiente para este último caso). Deben aclarar que política utilizan y **justificar** por qué es válida.

> *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.***

Representación de la información y sistemas combinacionales y secuenciales

Representación de la información

Sean A y B dos números de punto flotante de 32 bits en el formato IEEE 754, con la siguiente forma: $A = s_A \cdot m_A \cdot 2^{E_A}$ y $B = s_B \cdot m_B \cdot 2^{E_B}$, donde s_A, s_B son los bits de signo, m_A, m_B son los bits de la mantisa, $E_A = e_A + bias, E_B = e_B + bias$ son los bits del exponente y $bias = 127$.

- **Se pide:** dar la expresión para la operación de multiplicación entre A y B. Es decir, siendo $C = A \cdot B$:
  - Ejemplo: $m_C = m_A \cdot m_B$.
  - s_C =?
  - E_C =?
  - Expresión general para la multiplicación: $C =$ _____ $\cdot(m_A \cdot m_B) \cdot 2$ ^{_____}

Sistemas combinacionales y secuenciales

Para ciertas aplicaciones de IA estamos interesados en implementar en hardware una unidad de punto flotante (FPU) que realice solamente la multiplicación de dos números.

- **Se pide:**
  - Diseñar el circuito que implementa la operación correspondiente entre los dos exponentes de los números de punto flotante.

  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.