Primer Recuperatorio Organización del Computador

FIUBA - 2025-1C

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

Assembly

Implementar `listAddFirst` en assembly que inserte un nuevo nodo al inicio de una lista doblemente enlazada. La función debe recibir un puntero a la lista y un puntero al dato a insertar.

```
void listAddFirst(list_t* l, void* data);
```

La estructura de la lista y del nodo es:

```
typedef struct s_list {
    type_t   type;
    uint8_t  size;
    struct s_listElem* first;
    struct s_listElem* last;
} list_t;

typedef struct s_listElem {
    void* data;
    struct s_listElem* next;
    struct s_listElem* prev;
} listElem_t;
```

Acá se enumeran las definiciones extras que son necesarias:

```
typedef enum e_type {
    TypeNone = 0,
    TypeInt = 1,
    TypeString = 2,
    TypeCard = 3
} type_t;

typedef void* (funcClone_t)(void*);
funcClone_t* getCloneFunction(type_t t);
```

Condiciones a cumplir:

- El dato debe clonarse utilizando la función de clonación correspondiente al tipo de dato de la lista.
- El dato que se pasa siempre es válido.
- La lista que se pasa siempre es válida, pero puede tener tamaño 0.
- No chequear si `malloc` falla, se asume que siempre se puede alocar memoria.
- La función debe ser implementada en assembly.
- Se puede llamar a funciones de la biblioteca estándar de C.
- No se permite llamar a otras funciones de C que no figuren en el enunciado.
- Se evaluará el uso correcto de la pila, el cumplimiento de la ABI y el manejo de memoria dinámica.

Microarquitectura

Queremos expandir la capacidad de la arquitectura `OrgaSmall` en cuanto a operaciones aritméticas.

Se pide:

- Expandir la ISA `OrgaSmall` agregando la instrucción `NEG` que obtenga el inverso aditivo de un número. La misma debe modificar los flags.

Representación de la información

Se tienen los siguientes operandos representados en formato de punto flotante de precisión simple (IEEE 754):

- $A = 1.0$ y $B = b \times 2^{-25}$

donde b es la mantisa de B, y B es un flotante normalizado.

Se pide:

- Explicar con un esquema cómo se realiza la suma de $A + B$ en el formato IEEE 754.
- Dar el resultado exacto de dicha suma.

Justificar clara y concisamente la respuesta.

Memoria cache

Se dispone de una pequeña imagen en memoria de 16 pixels x 16 pixels, donde cada pixel se representa como:

```
typedef struct {
    uint8_t R; // Componente rojo
    uint8_t G; // Componente verde
    uint8_t B; // Componente azul
    uint8_t A; // Componente alfa
} pixel;
```

Y la imagen se almacena en una matriz de pixels de la siguiente manera:

```
pixel imagen[16][16]; // Matriz de pixels
```

Se quiere calcular el brillo promedio de la imagen, considerando el brillo de cada pixel como la media aritmética de sus componentes R, G y B. Para ello, se dispone de una función que recibe la imagen y devuelve el brillo promedio. La parte principal de la función es la siguiente:

```
    ...
    float suma_brillo = 0.0;
    int total_pixels = 16 * 16; // Total de pixels en la imagen

    for (int i = 0; i < 16; i++) {
        for (int j = 0; j < 16; j++) {
            pixel p = imagen[i][j];
            float brillo_pixel = (p.R + p.G + p.B) / 3.0;
            suma_brillo += brillo_pixel;
        }
    }

    float brillo = suma_brillo / total_pixels;
    ...
}
```

Se tiene una memoria caché con las siguientes características:

- 128 bytes de tamaño total.
- Mapeo directo.
- Tamaño de bloque de 16 bytes.

Suponer:

- Salvo la imagen el resto de las variables(`i, j, suma_brillo, total_pixels`) ya se encuentran alojadas en registros.
- La cache está vacía al inicio.
- La imagen se almacena en row-major order, es decir, los pixels de la fila 0 están en las primeras posiciones de memoria, seguidos por los de la fila 1, y así sucesivamente.

Responder:

1. ¿Cuál es el hit rate de la caché al ejecutar el código anterior? Justificar la respuesta.
2. ¿Cuál es el miss rate?
3. Si cambiamos `i` por `j`, es decir:

```
for (int i = 0; i < 16; i++) {
    for (int j = 0; j < 16; j++) {
        pixel p = imagen[j][i];
        float brillo_pixel = (p.R + p.G + p.B) / 3.0;
        suma_brillo += brillo_pixel;
    }
}
```

¿Cuál es el hit rate de la caché ahora? Justificar la respuesta.

Paginación

Necesitamos alocar en memoria las siguientes estructuras de datos en una arquitectura de 32 bits::

```
int main(void) {
    ...
    double *array = malloc(1024 * sizeof(double));
    int32_t data[1024];
    int32_t moreData[1024];
    ...
}
```

Sabemos que tanto el heap como el stack tienen menos de una página ocupada.

Además considerar:

- Las estructuras de paginación se encuentran en las siguientes direcciones físicas:
  - Page Directory: 0x00001000
  - Page Table: 0x00004000
- El heap debe ocupar posiciones de memoria física contiguas (esto no es siempre el caso, pero para este ejercicio lo asumimos) y su mapeo inicial es: Virtual 0x00010000 a Física 0x04000000.
- El stack debe ocupar posiciones de memoria física contiguas (esto no es siempre el caso, pero para este ejercicio lo asumimos) y su mapeo inicial es: Virtual 0xFFFFFFFF a Física 0x03FFFFFF.
- El contenido de las estructuras deben ocupar páginas nuevas, asuminedo que si se necesita memoria extra se puede utilizar las páginas alocadas previamente.

Se pide:

- Completar las estructuras de paginación (PD y PT) para que las estructuras de datos queden correctamente alocadas en memoria.

Page Directory:

0x00001:

Entry	Page Table
0	0x00004
1023	0x00004

Page Tables:

0x00004:

Entry	Page
16	0x04000
1023	0x03FFF