Ejercicios

Ejercicio 1

Dada una matriz de *nxn* enteros de 16 bits, programar una rutina que devuelva los elementos de la diagonal en el vector pasado por parámetro.
El prototipo de la función es:

```
void diagonal(short* matriz, short n, short* vector);
```

Ejercicios

Ejercicio 1 - Solución

```
diagonal:
    xor rcx, rcx
    mov cx, si                  ;iteraciones = filas
    mov rsi, rcx                ;limpiamos los parametros
                                ;buscamos los elementos
                                  de la diagonal
.ciclo:
    mov r8w, [rdi]              ;guardamos el elemento
    mov [rdx], r8w

    lea rdi, [rdi+2*rsi+2]      ;nos movemos hasta el siguiente
                                  elemento de la diagonal
    lea rdx, [rdx+2]            ;nos movemos un lugar
                                  en el vector
    loop .ciclo
.fin:
    ret
```

Ejercicios

Ejercicio 2

Dada una matriz de *nxn* enteros de 16 bits, programar una rutina que devuelva los elementos de la diagonal en un vector **nuevo**.
El prototipo de la función es:

```
short* diagonal(short* matriz, short n);
```

- ¿Qué implica la frase "devolver la diagonal en un vector nuevo"?

Ejercicios

Ejercicio 2

Dada una matriz de *nxn* enteros de 16 bits, programar una rutina que devuelva los elementos de la diagonal en un vector **nuevo**.
El prototipo de la función es:

```
short* diagonal(short* matriz, short n);
```

- ¿Qué implica la frase "devolver la diagonal en un vector nuevo"? Que tenemos que crear un vector.
- ¿Cómo lo hacemos?

Ejercicios

Ejercicio 2

Dada una matriz de *nxn* enteros de 16 bits, programar una rutina que devuelva los elementos de la diagonal en un vector **nuevo**.
El prototipo de la función es:

```
short* diagonal(short* matriz, short n);
```

- ¿Qué implica la frase "devolver la diagonal en un vector nuevo"?
  Que tenemos que crear un vector.
- ¿Cómo lo hacemos?
  Pedimos memoria, asi:

```
void * malloc (size_t size)
```

donde `size` es la cantidad de bytes que queremos pedir.
El valor de retorno es un puntero al espacio reservado.

Ejercicios

Ejercicio 2 - Solución

```
diagonal:
    ;en rdi tengo la matriz
    ;en si tengo el tamaño
     de la matriz
                                xor rcx, rcx
                                mov cx, si
                                mov rsi, rcx

                            .ciclo:
                                mov r8w, [rdi]
                                mov [rax], r8w

                                lea rdi, [rdi+2*rsi+2]
                                lea rax, [rax+2]
                                loop .ciclo
                                mov rax, r15

                            .fin:
                                pop r15
                                pop rbp
                                ret
```

Ejercicios

Ejercicio 2 - Solución

```
diagonal:
    ;en rdi tengo la matriz
    ;en si tengo el tamaño
     de la matriz

    push rbp
    mov rbp, rsp
    push r15
    push rdi
    push rsi

    xor rdi, rdi
    mov di, si
    shl rdi, 1
    sub rsp, 8
    call malloc
    add rsp, 8
    mov r15, rax

    pop rsi
    pop rdi
    xor rcx, rcx
    mov cx, si
    mov rsi, rcx

.ciclo:
    mov r8w, [rdi]
    mov [rax], r8w

    lea rdi, [rdi+2*rsi+2]
    lea rax, [rax+2]
    loop .ciclo
    mov rax, r15

.fin:
    pop r15
    pop rbp
    ret
```

Ejercicios

Ejercicio 3

Dado un vector de n enteros de 16 bits, programar una rutina que devuelva la suma de los elementos del vector y luego lo **destruya**.
El prototipo de la función es:

```
short suma(short* vector, short n);
```

- ¿Qué implica la frase "destruír el vector"?

Ejercicios

Ejercicio 3

Dado un vector de n enteros de 16 bits, programar una rutina que devuelva la suma de los elementos del vector y luego lo **destruya**.
El prototipo de la función es:

```
short suma(short* vector, short n);
```

- ¿Qué implica la frase "destruír el vector"?
  Que básicamente tenemos que romper todo el vector.
  Más formal: Devolver al sistema la memoria que ocupaba el vector y que ya no vamos a usar.
- ¿Cómo lo hacemos?

Ejercicios

Ejercicio 3

Dado un vector de n enteros de 16 bits, programar una rutina que devuelva la suma de los elementos del vector y luego lo **destruya**.
El prototipo de la función es:

```
short suma(short* vector, short n);
```

- ¿Qué implica la frase "destruír el vector"?
  Que básicamente tenemos que romper todo el vector.
  Más formal: Devolver al sistema la memoria que ocupaba el vector y que ya no vamos a usar.
- ¿Cómo lo hacemos?
  Devolvemos (liberamos) la memoria que ocupa asi:

```
void free ( void * puntero )
```

que toma un puntero al espacio de memeoria que queremos liberar.
Advertencia: Una vez que llamamos a `free` ya **no hay vuelta atrás**.
Todo lo que intentemos hacer con el vector luego de la llamada resultará en un error ya que esa memoria dejó de ser nuestra.

Ejercicios

Ejercicio 3 - Solución

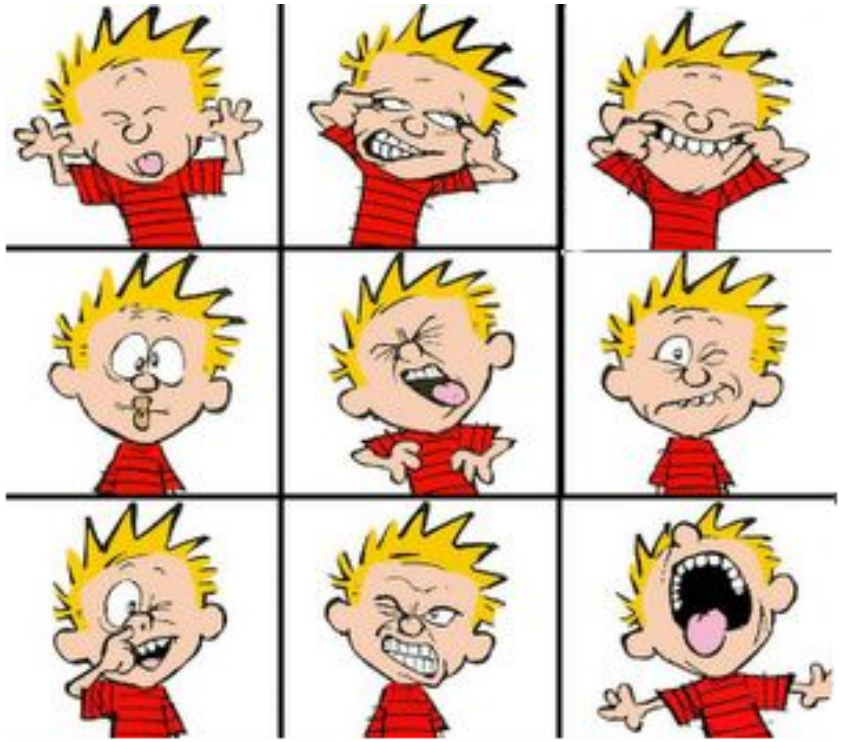
```
suma:
    ;en rdi tengo el vector
    ;en si tengo el tamaño
     push rbp
     mov rbp, rsp
     push rbx
     push r12
     mov rbx, rdi

     xor r12,r12
     xor rcx, rcx
     mov cx, si
```

```
.cicloSuma:
     add r12w, [rdi]
     lea rdi, [rdi+2]
     loop .cicloSuma

     mov rdi, rbx
     call free
     mov rax, r12
.fin:
     pop r12
     pop rbx
     pop rbp
     ret
```

¿Preguntas?



[1]

[1] Calvin & Hobbes by Bill Watterson

A trabajar!

Ejercicios 1, 2 y 3

Implementar y probar los ejercicios 1 y 2 utilizando la digonal secundaria y el ejercicio 3 con la operación resta.

Ejercicio 4

Dada una matriz de *nxn* enteros sin signo de 16 bits y un vector de longitud *n* ($n > 1$ es de 16 bits). Programar una rutina que rellene el vector de manera que en la posición *i* se encuentre la sumatoria de los elementos de la fila *i* de la matriz.
El prototipo de la función es:

```
void diagonal(unsigned short int* matriz, short n,
                    unsigned short int* vector);
```