

Práctica 3: Programación en ASM "Orga 1"

Organización del Computador I
DC - UBA

2^{do} Cuatrimestre 2017

Ejercicio 1

Enunciado:

Gus nos pidió escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

Ejercicio 1

Enunciado:

Gus nos pidió escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

- R1 contiene la dirección de memoria donde se aloja el dividendo.

Ejercicio 1

Enunciado:

Gus nos pidió escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

- R1 contiene la dirección de memoria donde se aloja el dividendo.
- R2 contiene la dirección de memoria donde se aloja el divisor.

Ejercicio 1

Enunciado:

Gus nos pidió escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

- R1 contiene la dirección de memoria donde se aloja el dividendo.
- R2 contiene la dirección de memoria donde se aloja el divisor.
- R3 debe ser el registro en el que se devuelva el cociente (el resultado de la división).

Ejercicio 1

Enunciado:

Gus nos pidió escribir un programa que calcule la división entera entre dos enteros sin signo de 16 bits.

- R1 contiene la dirección de memoria donde se aloja el dividendo.
- R2 contiene la dirección de memoria donde se aloja el divisor.
- R3 debe ser el registro en el que se devuelva el cociente (el resultado de la división).
- En caso de que el divisor sea 0, habrá que devolver 0.

Pseudocódigo del Ejercicio 1

```
resultado = 0
if (divisor == 0):
    listo
else:
    while (dividendo >= divisor):
        dividendo = dividendo - divisor
        resultado = resultado + 1
    listo
```

Resolución del Ejercicio 1

```
; R1 --> puntero al dividendo
; R2 --> puntero al divisor
; R3 --> cociente
; R4 --> dividendo
; R5 --> divisor

inicio: MOV R3, 0x0000 ; R3 = 0
        MOV R4, [R1]      ; R4 = dividendo
        MOV R5, [R2]      ; R5 = divisor
        CMP R5, 0x0000 ; divisor == 0?
        JE fin

ciclo:  CMP R4, R5      ; dividendo < divisor?
        JCS fin          ; uso JCS en lugar de JL porque
                           ; son enteros sin signo
        SUB R4, R5      ; R4 = R4-R5
        ADD R3, 0x0001 ; R3 = R3+1
        JMP ciclo

fin:
```

Ejercicio 2

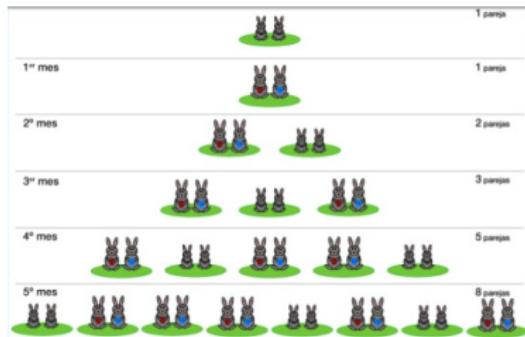
Enunciado:

Vero compró una pareja de conejos y nos pidió ayuda para saber cuántos se podrían reproducir en un año a partir de la pareja inicial, teniendo en cuenta que de forma natural tienen una pareja en un mes, y que a partir del segundo se empiezan a reproducir.

Ejercicio 2

Enunciado:

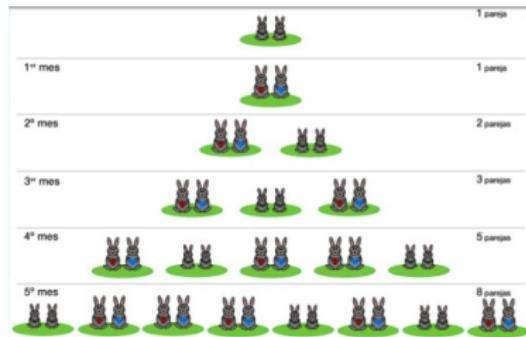
Vero compró una pareja de conejos y nos pidió ayuda para saber cuántos se podrían reproducir en un año a partir de la pareja inicial, teniendo en cuenta que de forma natural tienen una pareja en un mes, y que a partir del segundo se empiezan a reproducir.



Ejercicio 2

Enunciado:

Vero compró una pareja de conejos y nos pidió ayuda para saber cuántos se podrían reproducir en un año a partir de la pareja inicial, teniendo en cuenta que de forma natural tienen una pareja en un mes, y que a partir del segundo se empiezan a reproducir.



Sucesión de Fibonacci

$$f_n = f_{n-1} + f_{n-2} \text{ con } f_0 = 1 \text{ y } f_1 = 1$$

Pseudocódigo del Ejercicio 2

```
meses = 10;      // (12-2) me da el termino 12 de la suc.  
anteUltMes = 1;  
ultMes = 1;  
do{ cantParejas = ultMes;  
    cantParejas = cantParejas + anteUltMes;  
    anteUltMes = ultMes;  
    ultMes = cantParejas;  
    meses = meses - 1;  
}while (meses > 0)
```

Resolución del Ejercicio 2

```
; R0 --> meses
; R1 --> anteUltMes
; R2 --> ultMes
; R3 --> cantParejas

inicio: MOV R0, 0x000A ; meses = 10
        MOV R1, 0x0001 ; anteUltMes = 1
        MOV R2, 0x0001 ; ultMes = 1
ciclo:  MOV R3, R2      ; cantParejas = ultMes;
        ADD R3, R1      ; cantParejas += anteUltMes;
        MOV R1, R2      ; anteUltMes = ultMes;
        MOV R2, R3      ; ultMes = cantParejas;
        SUB R0, 0x0001 ; meses = meses - 1;
        JG  ciclo       ;
```

Ejercicio 3

Enunciado:

Leo está cansado de que las fotos le salgan oscuras con su cámara digital Nikonflash. Nos pidió que hagamos un programa que duplique el brillo de las fotos para remediar su angustia. Por cierto, la cámara de Leo es viejita, sólo imprime en blanco y negro y tiene una resolución de 200x200 píxeles.

Ejercicio 3

Enunciado:

Leo está cansado de que las fotos le salgan oscuras con su cámara digital Nikonflash. Nos pidió que hagamos un programa que duplique el brillo de las fotos para remediar su angustia. Por cierto, la cámara de Leo es viejita, sólo imprime en blanco y negro y tiene una resolución de 200x200 píxeles.

- R1 contiene la dirección donde está la imagen como una matriz de enteros sin signo de 16 bits.

Ejercicio 3

Enunciado:

Leo está cansado de que las fotos le salgan oscuras con su cámara digital Nikonflash. Nos pidió que hagamos un programa que duplique el brillo de las fotos para remediar su angustia. Por cierto, la cámara de Leo es viejita, sólo imprime en blanco y negro y tiene una resolución de 200x200 píxeles.

- R1 contiene la dirección donde está la imagen como una matriz de enteros sin signo de 16 bits.
- La foto tiene 200x200 pixels.

Ejercicio 3

Enunciado:

Leo está cansado de que las fotos le salgan oscuras con su cámara digital Nikonflash. Nos pidió que hagamos un programa que duplique el brillo de las fotos para remediar su angustia. Por cierto, la cámara de Leo es viejita, sólo imprime en blanco y negro y tiene una resolución de 200x200 píxeles.

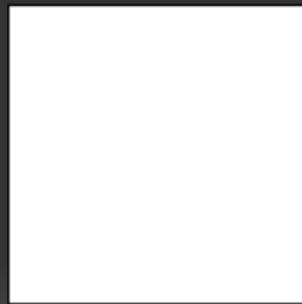
- R1 contiene la dirección donde está la imagen como una matriz de enteros sin signo de 16 bits.
- La foto tiene 200x200 pixels.
- La imagen debe ser modificada en el lugar.

Pixels

0x0000



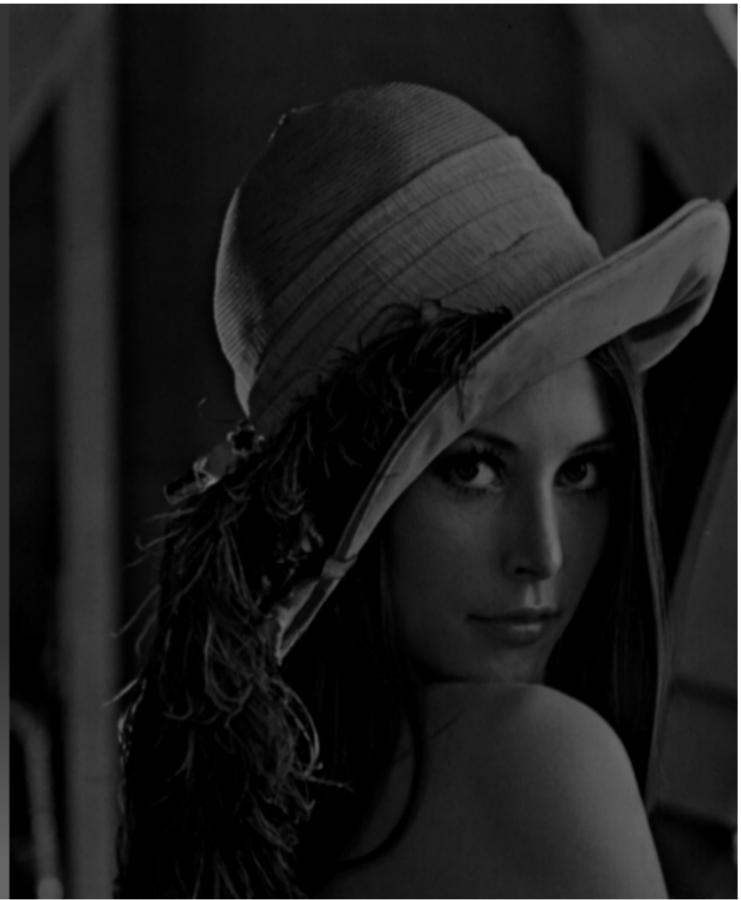
0xFFFF



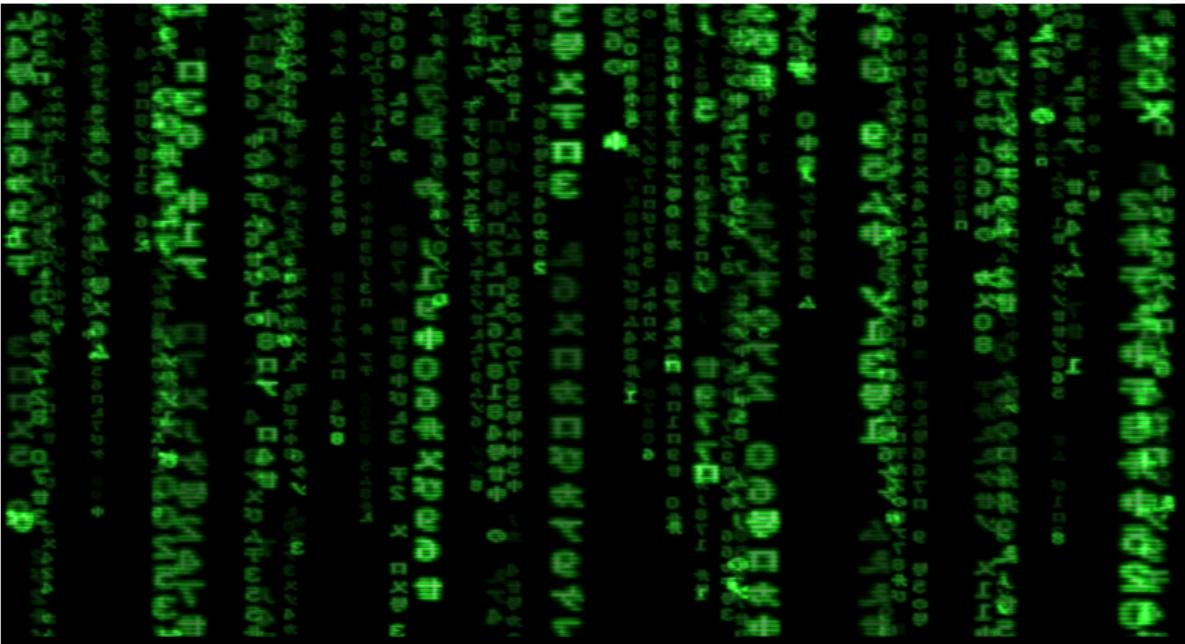
Lena

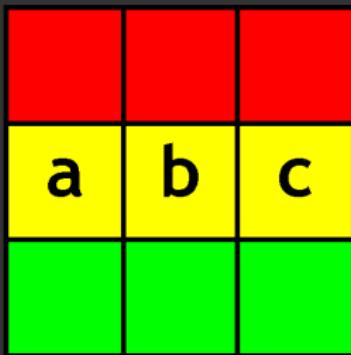


Lena



¿Matriz?





4	
5	
6	
a	7
b	8
c	9
10	
11	
12	

Pseudocódigo del Ejercicio 3

Enunciado

Leo está cansado de que las fotos le salgan oscuras con su cámara digital Nikonflash. Nos pidió que hagamos un programa que duplique el brillo de las fotos para remediar su angustia. Por cierto, la cámara de Leo es viejita, sólo imprime en blanco y negro y tiene una resolución de 200x200 píxeles.

Pseudocódigo del Ejercicio 3

Enunciado

Leo está cansado de que las fotos le salgan oscuras con su cámara digital Nikonflash. Nos pidió que hagamos un programa que duplique el brillo de las fotos para remediar su angustia. Por cierto, la cámara de Leo es viejita, sólo imprime en blanco y negro y tiene una resolución de 200x200 píxeles.

```
i = 0
while (i < 200*200):
    img[i] = 2 * img[i]
    i = i + 1
listo
```

¿Es correcto el pseudocódigo
anterior?

¿Es correcto el pseudocódigo anterior?

¿Qué pasa cuando duplico un pixel cuyo color ya es muy cercano al blanco más 'brillante'?

Pseudocódigo del Ejercicio 3 (con saturación)

```
i = 0
while (i < 200*200):
    tmp = 2 * img[i]
    if (tmp > 0xFFFF):           // si hay carry
        tmp = 0xFFFF
    img[i] = tmp
    i = i + 1
listo
```

Resolución del Ejercicio 3

```
; R1 --> puntero a la imagen
; R2 --> variable temporal para almacenar el pixel a procesar
; R3 --> contador de pixeles que voy procesando

inicio: MOV R3, 0x0000 ; R3 = 0

ciclo:  CMP R3, 0x9C40 ; R3 = cant pixeles(img)?
        JE fin
        MOV R2, [R1]      ; R2 = img[R3]
        ADD R2, R2        ; R2 = 2*R2

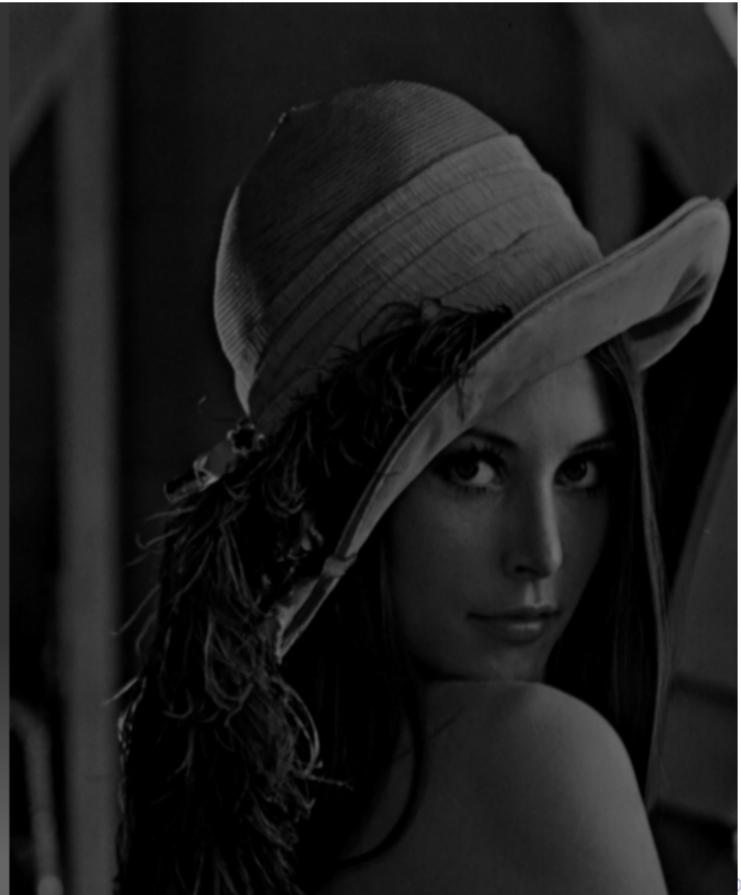
; Como estoy sumando enteros de 16 bits sin signo, tengo que ver que
; no se produzca overflow. Para eso, miro el bit de carry despues de
; realizada la suma

        JCS saturar
cont:   MOV [R1], R2      ; img[R3] = R2
        ADD R1, 0x0001    ; R1 = img[R3+1]
        ADD R3, 0x0001    ; R3 = R3+1
        JMP ciclo

saturar:
; Si no entra en un entero de 16 bits, saturo el valor del pixel
; con el color blanco (0xFFFF)
        MOV R2, 0xFFFF    ; R2 = 0xFFFF
        JMP cont

fin:
```

Orig.



x2



x2
(con saturación)



?

Fin.

