Organización de computadoras

Clase 5

Universidad Nacional de Quilmes

Lic. Martínez Federico

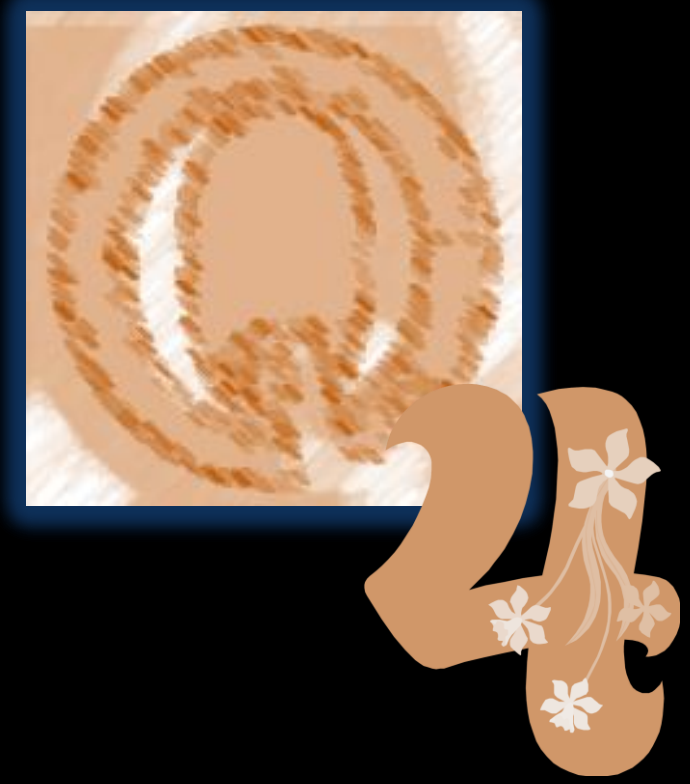
El Parcial

Repasando

[[0x0000]]

[R_0]

Arreglos




AQUÍ
Y
AHORA

¿Y ahora?

- Pila (porque no se entendió)

¿Y ahora?

- Pila (porque no se entendió)
- Modularización

¿Y ahora?

- Pila (porque no se entendió)
- Modularización
- Reuso

¿Y ahora?

- Pila (porque no se entendió)
- Modularización
- Reuso
- Especificación por contratos

¿Y ahora?

- Pila (porque no se entendió)
- Modularización
- Reuso
- Especificación por contratos
- Llamadas a subrutinas

¿Y ahora?

- Pila (porque no se entendió)
- Modularización
- Reuso
- Especificación por contratos
- Llamadas a subrutinas
- Q5

Pila

Pila

- Push: Agrega un elemento en el tope de la pila

Pila

- Push: Agrega un elemento en el tope de la pila
- Pop: Saca el elemento del tope

Stack pointer

- Registro interno

Stack pointer

- Registro interno
- Apunta al tope de la pila (Dirección)

Stack pointer

SP →

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x0000
0xFFF3	0x0A53
0XFFF4	0x1111

Stack pointer

SP →

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x0000
0xFFF3	0x0A53
0XFFF4	0x1111

Pop

Stack pointer

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x0000
0xFFF3	0x0A53
0XFFF4	0x1111

SP ⟶

Pop
0x0A53

Stack pointer

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x0000
0xFFF3	0x0A53
0XFFF4	0x1111

SP →

Push 0xFEDE

Stack pointer

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x0000
0xFFF3	0xFEDE
0XFFF4	0x1111

SP →

Push 0xFEDE

Stack pointer

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x0000
0xFFF3	0xFEDE
0XFFF4	0x1111

SP →

Push 0x78D3

Stack pointer

SP 

Dirección	Contenido
0xFFF0	0x0000
0xFFF1	0x0000
0XFFF2	0x78D3
0xFFF3	0xFEDE
0XFFF4	0x1111

Push 0x78D3

Pila

- La arquitectura decide el valor inicial del SP
- La memoria que ocupa la pila es memoria común y corriente
- La máquina no “válida” si al pushear no se pierden algún dato

Ejercicio

- Sumar los dos números que están en la pila y devolver el resultado en la misma

Ejercicio

- Sumar los dos números que están en la pila y devolver el resultado en la misma

```
POP R0
POP R1
ADD R1, R0
PUSH R1
```

Ejercicio

- Se tiene un arreglo que comienza en la dirección que se encuentra en R0. Se pide dar vuelta el arreglo

Dirección	Contenido
0x00E0	0x0001
0x00E1	0x111F
0x00E2	0x322A
0x00E3	0x5BDC

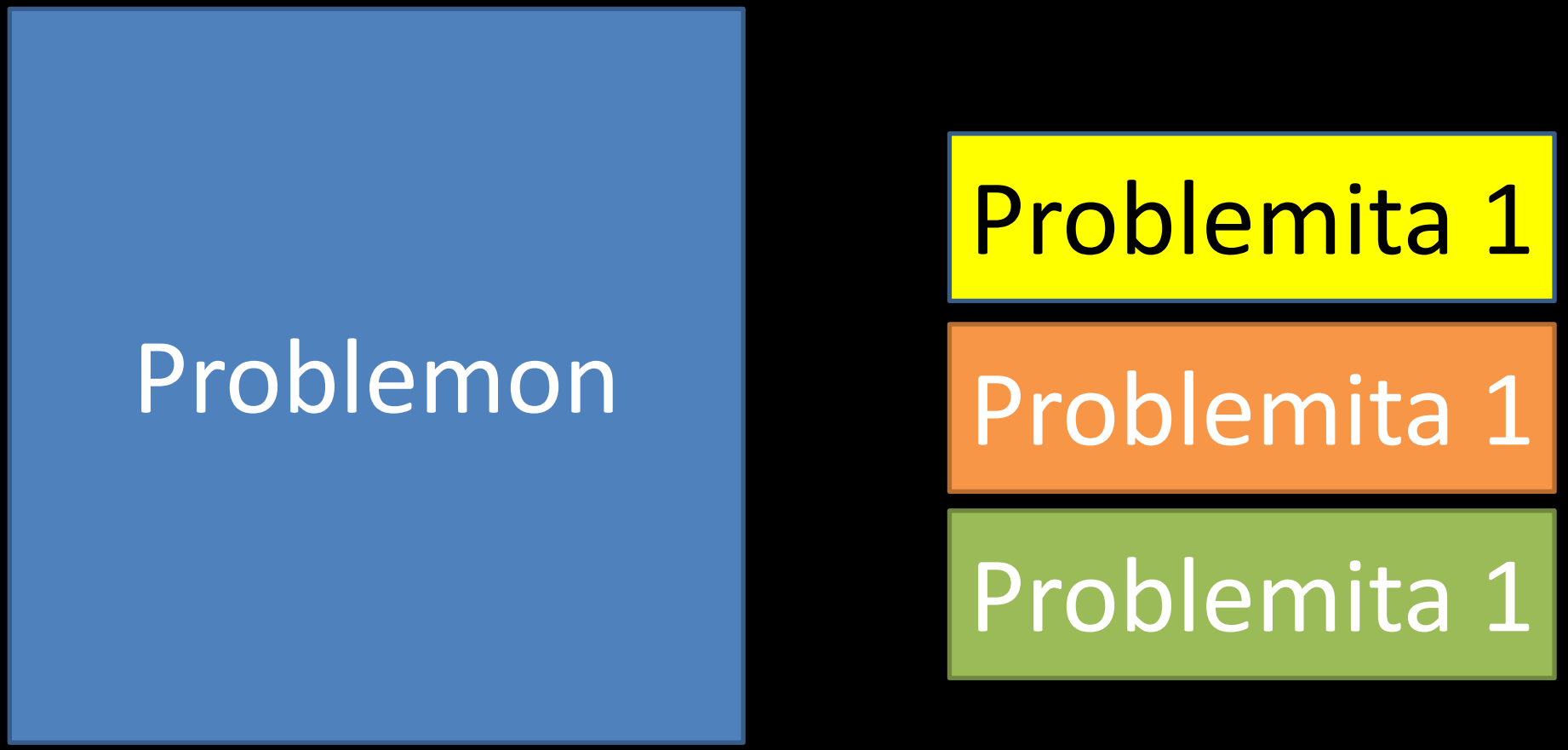
Dirección	Contenido
0x00E0	0x5BDC
0x00E1	0x322A
0x00E2	0x111F
0x00E3	0x0001

Hint: la pila

Modularizar

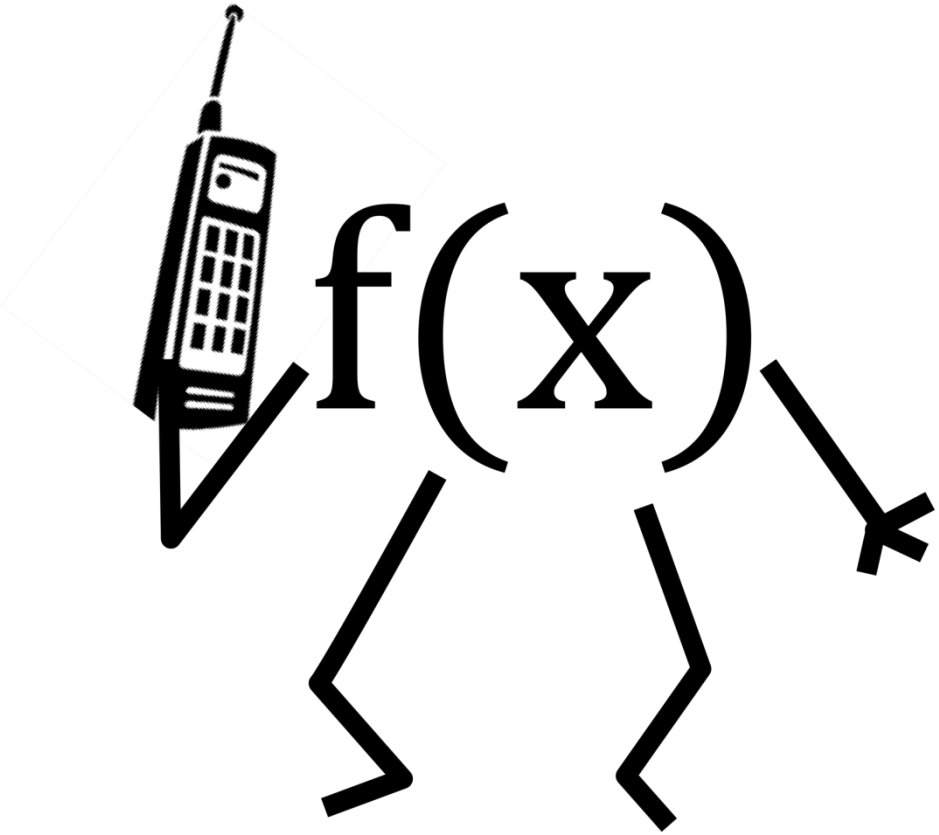
Modularizar

- Partir un problema en varios mas chicos



Modularizar

- Usar llamadas a función



Ejemplo

- Hacer un programa que cuente cuantos números de un arreglo son pares, tienen el bit 5 en 1 y son múltiplos de 5

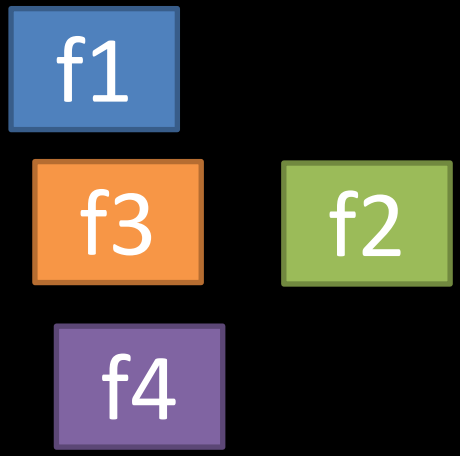
Ejemplo

- Hacer un programa que cuente cuantos números de un arreglo son pares, tienen el bit 5 en 1 y son múltiplos de 5

Reuso

Reuso

- Escribir funciones que puedan ser usadas para resolver otros problemas. Y aprovecharlas!



Problema 1

Problema 2

Reuso

- Ejercicios 7 y 8 del parcial

Especificación por contratos

Especificación por contratos

- ¿Por qué?
  - Queremos poder reutilizar subrutinas
  - No podemos andar leyendo cada una para saber donde hay que pasarle los parámetros
  - Ni para saber que hacen
  - Lo documentamos!

Especificación por contratos

- Modelamos con:
  - Requiere
  - Retorna
  - Modifica

Especificación por contratos

- Pedimos lo que necesitamos: Requiere
- Notificamos qué vamos a devolver: Retorna
- Informamos que registros, celdas de memoria, flags, etc; vamos a modificar: Modifica

Requiere

- Dónde están los operandos?

- Ej: En R0 la dirección de comienzo del array y en R1 el tamaño

Requiere

- Dónde están los operandos?
- Ej: En R0 la dirección de comienzo del array y en R1 el tamaño
- Qué características deben tener?
- Ej: Ningún numero de un array es 0, la suma de los dos operandos no se va de rango, etc

Retorna

- Cuál es el valor (o valores) que retorna y donde.

- Ej: En R0 devuelve la suma de los valores del array. En R1 devuelve 1 si y solo si el número que estaba en R2 era par.

Modifica

- ¿Cambia el valor de algún registro?
- ¿Cambian celdas de memoria?
- ¿Se modifican los flags?

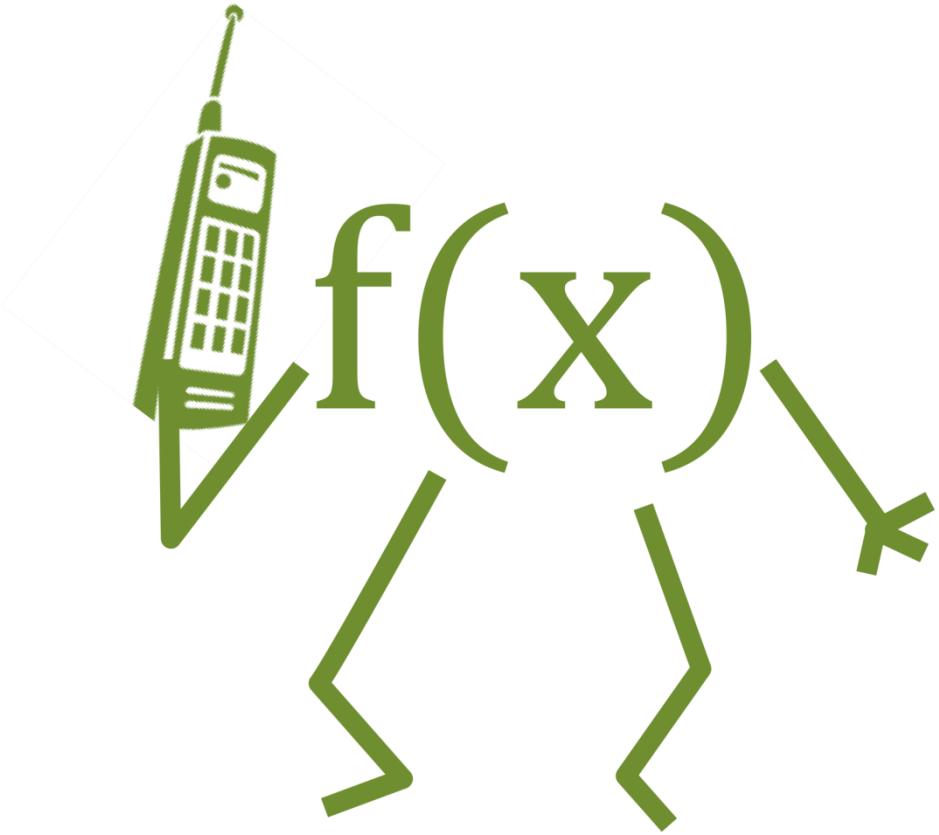
Ejemplo

- Hacer una subrutina que reciba un número en BSS(16) en R0 y devuelva en R1 un 1 si es mayor a 10 y un 0 si no

Ejemplo

- Hacer un programa que dado un arreglo de números en BSS(16) que comienza en la posición cuya dirección está en R0 y su tamaño en R1, devuelva el mínimo.

Llamadas a subrutinas



Llamadas a subrutinas

```
esPar: and R0, 0x0001
       JNE impar
       mov R1, 0x0001
       jmp fin
Impar: mov R1, 0x0000
fin
```

Llamadas a subrutinas

- Queremos dejar de ejecutar el código actual y pasar a ejecutar el de la subrutina

Llamadas a subrutinas

- Queremos dejar de ejecutar el código actual y pasar a ejecutar el de la subrutina

- ¿Podemos usar un JMP?


¡¡No!!

¡¡¡¡¡NO!!!!

Si usamos JMP, ¿Cómo volvemos?*

*Se puede utilizar jmp para un caso particular, pero no permite el reuso

CALL y RET

- Necesitamos nuevas instrucciones para poder ir y volver

CALL y RET

- CALL: salta, pero guarda en la pila la dirección a donde tiene que volver

- RET: Salta al valor que está en el tope del stack

CALL y RET

- 0x00F0 Rutina1: mov r0, r1
  call rutina2
  ret

- 0x00A1 Rutina2: mov r2,r3
  ret

- 0x0E00 Principal: call Rutina2
  call Rutina1

¿Cómo varía la pila?

CALL y RET

¿Qué pasa si la rutina no hace ret?

Sigue corriendo!!

CALL y RET

¿Qué pasa si la rutina no hace ret?

¿Qué pasa si la subrutina saca el valor de retorno de la pila?

CALL y RET

¿Qué pasa si la rutina no hace ret?

¿Qué pasa si la subrutina saca el valor de retorno de la pila?

¿Qué pasa si la subrutina pone cosas en la pila y no las saca?


Q
5

- Modos de direccionamiento

Modo	Código
Inmediato	000000
Registro	100RRR
Directo	001000
Indirecto	011000
Registro Indirecto	110RRR

- Formato de instrucción:
  - Instrucciones tipo 1:

Cod Op (4bits)	Modo Destino (6 bits)	Modo origen (6 bits)	Destino (16 bits)	Origen (16 bits)

Operación	Código	Efecto
MUL	0000	Dest ←Dest * Origen
MOV	0001	Dest← Origen
ADD	0010	Dest ← Dest + Origen
SUB	0011	Dest← Dest - Origen
DIV	0111	Dest← Dest% Origen
CMP	0110	Modifica los Flags según el resultado de Dest – Origen
AND	0100	Dest ← Dest AND Origen
OR	0101	Dest ← Dest OR Origen

- Formato de instrucción:
  - Instrucciones tipo 2:

Cod Op (4 bits)	Relleno (000000)	Modo Origen (6 bits)	Origen (16 bits)

- Operaciones tipo 2:

Operación	Código	Efecto
JMP	1010	PC ← Origen
PUSH	1110	[SP] ← Origen; SP← SP – 1
CALL	1011	[SP] ←PC; SP←SP-1; PC ←Origen

- Formato de instrucción:
  - Instrucciones tipo 3:

Prefijo (1111)	Cod Op (4 bits)	Desplazamiento (8 bits)

- Operaciones tipo 3:

Salto	Codop	Descripción	Condición
JE	0001	Igual / Cero	Z
JNE	1001	No igual	¬ Z
JLE	0010	Menor o igual con signo	Z + (N ⊕ V)
JG	1010	Mayor con signo	¬(Z + (N ⊕ V))
JL	0011	Menor con signo	N ⊕ V
JGE	1011	Mayor o igual con Signo	¬ (N ⊕ V)
JLEU	0100	Menor o igual sin signo	C + Z
JGU	1100	Mayor sin signo	¬(C + Z)
JCS	0101	Menor sin signo	C
JNEG	0110	Negativo	N
JVS	0111	Overflow	V

- Formato de instrucción:
  - Instrucciones tipo 4:

Cod Op (4 bits)	Modo Destino (6 bits)	Relleno (000000)	Destino (16 bits)

- Operaciones tipo 4:

Operación	Código	Efecto
NOT	1001	Dest ← NOT Dest
POP	1101	SP← SP + 1; Dest ← [SP]

- Formato de instrucción:
  - Instrucciones tipo 5:

Cod Op (4 bits)	Relleno (000000000000)

- Operaciones tipo 4:

Operación	Código	Efecto
RET	1100	PC ← [SP+1]; SP ← SP + 1

Ejercicio

- Se tienen dos arreglos que comienzan en las direcciones que se encuentran en R0 y R1. Ambos terminan con 0. Se quiere saber cuantos elementos del primer arreglo están en el otro.

Hint: Considere una subrutina Pertenece que dado un elemento diga si está en un arreglo

Ejercicio

- Se tienen 2 arreglos de números distintos. Uno comienza en la dirección que guarda R0 y otro en la de R1. Ambos terminan con 0. En R2 se tiene la dirección de un arreglo donde debe cargarse la unión sin repetidos de ambos arreglos, finalizando con un 0.

- Hint: ¿Puede reutilizar pertenece?

Qué deberían llevarse en la cabeza

Qué deberían llevarse en la cabeza

- Pila
  - SP
  - Push y Pop

Qué deberían llevarse en la cabeza

- Pila
  - SP
  - Push y Pop
- Subrtuinas
  - Herramienta para modularizar y reusar

Qué deberían llevarse en la cabeza

- Pila
  - SP
  - Push y Pop
- Subrtuinas
  - Herramienta para modularizar y reusar
- Contratos
  - Documentación de subrutinas
  - Requiere, Asegura y Modifica

Qué deberían llevarse en la cabeza

- Pila
  - SP
  - Push y Pop
- Subrtuinas
  - Herramienta para modularizar y reusar
- Contratos
  - Documentación de subrutinas
  - Requiere, Asegura y Modifica
- Call y Ret
  - ¿Qué?
  - Diferencias con JMP

Qué deberían llevarse en la cabeza

- Pila
  - SP
  - Push y Pop
- Subrtuinas
  - Herramienta para modularizar y reusar
- Contratos
  - Documentación de subrutinas
  - Requiere, Asegura y Modifica
- Call y Ret
  - ¿Qué?
  - Diferencias con JMP
- 

¿Preguntas?


GRACIAS!