Problemas Tema 3 – Bonus

# Programación en ensamblador

## Fundamentos de computadores II

**José Manuel Mendías Cuadros**
*Dpto. Arquitectura de Computadores y Automática*
*Universidad Complutense de Madrid*

# Ejercicio 18

## Pseudo-código

C/C++

```
#define N 5
int x = 4;
int y = 5;
int v[2*N] = { 1, 2, -3, 4, 5, 9, 17, -15, 20, 12 };
int d[N];
int abs( int x )
{
  if( x < 0 )
    x = -x;
  return  x;
}
int chebyshev( int x1, int y1, int x2, int y2 )
{
  int d1, d2;
  d1 = abs( x1-x2 );
  d2 = abs( y1-y2 );
  if( d2 > d1 )
    d1 = d2;
  return d1;
}
void main( void )
{
  int i;
  for( i = 0; i < N; i++ )
    d[i] = chebyshev( x, y, v[2*i], v[2*i+1] );
  while(1);
}
```

# Ejercicio 18

## Variables globales

C/C++

```
#define N 5

int x = 4;
int y = 5;
int v[2*N] = { 1,2,-3,4,5,9,17,-15,20,12 };
int d[N];
```

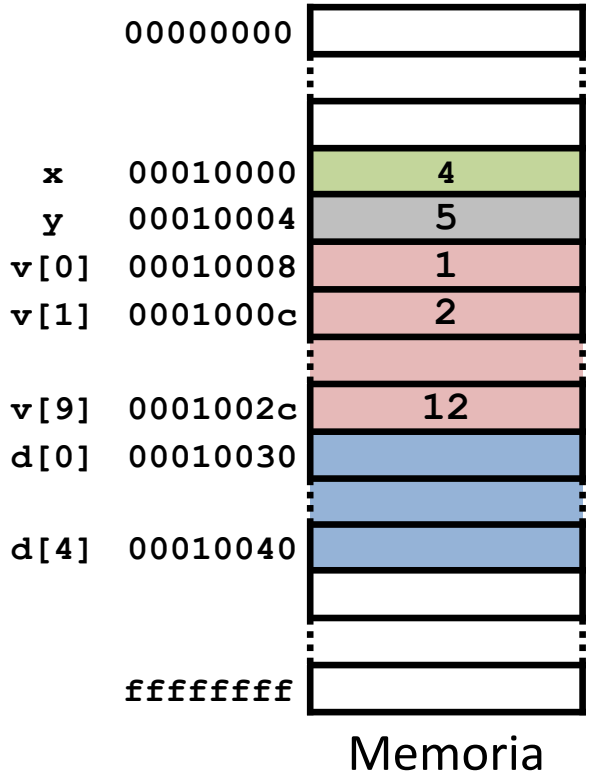
ASM

```
.global main
.extern _stack

.equ N, 5

.data
x: .word 4
y: .word 5
v: .word 1,2,-3,4,5,9,17,-15,20,12

.bss
d: .space N*4
```



# Ejercicio 18

## Programa principal (usando etiquetas)

C/C++

```
void main( void )
{
  int i;

  for( i = 0; i < N; i++ )
    d[i] = chebyshev(
              x,
              y,
              v[2*i],
              v[2*i+1] );

  while(1);
}
```

*i → s1, N → s2, v[] → s3, d[] → s4*

C/C++

```
main:
  la    sp, _stack
  mv    s1, zero
  li    s2, N
for:
  bge   s1, s2, efor
  la    t0, x
  lw    a0, 0(t0)
  la    t0, y
  lw    a1, 0(t0)
  la    s3, v
  sll   t0, s1, 1
  sll   t0, t0, 2
  add   t0, s3, t0
  lw    a2, 0(t0)
  lw    a3, 4(t0)
  call  chebyshev
  la    s4, d
  sll   t0, s1, 2
  add   t0, s4, t0
  sw    a0, 0(t0)
  add   s1, s1, 1
  j     for
efor:
  j     .
```

*Inicializa pila*

*Paso de parámetros*

*Calcula i*2*

*Calcula el desplazamiento*

*Suma base y desplazamiento*

*Carga* `v[2*i]`

*Carga* `v[2*i+1]`

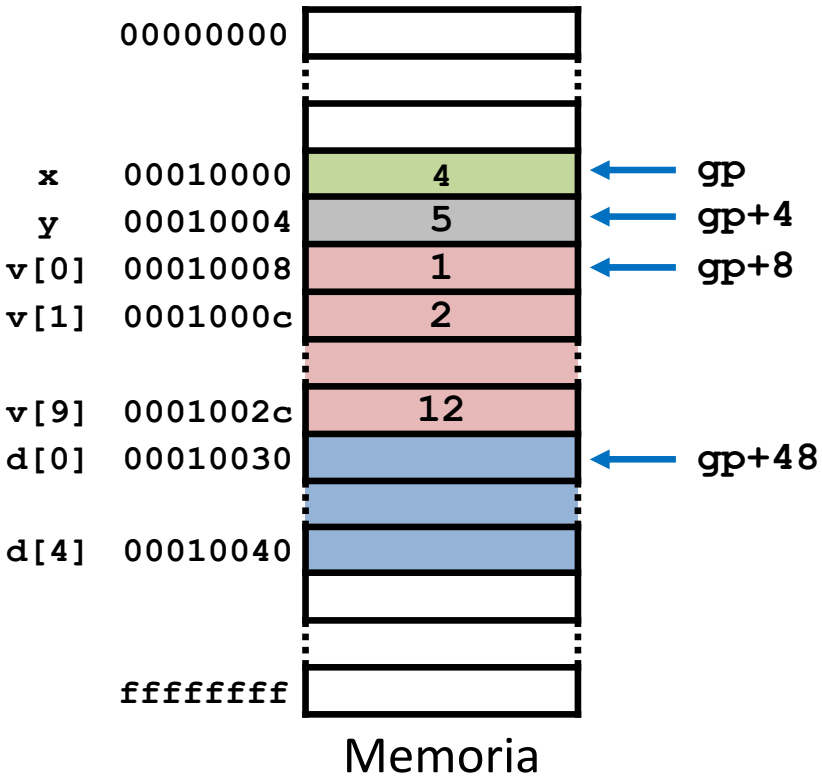
*Almacena resultado*

*Almacena en* `d[i]`

# Ejercicio 18

## Programa principal (usando gp)

*i → s1, N → s2, v[] → s3, d[] → s4*




C/C++

```
main:
  la   sp, _stack      <- Inicializa pila
  la   gp, x           <- Inicializa gp a la dirección de comienzo de las variables
  mv   s1, zero
  li   s2, N
for:
  bge  s1, s2, efor
  lw   a0, 0(gp)       <- Carga x
  lw   a1, 4(gp)       <- Carga y
  add  s3, gp, 8       <- Carga v[]
  sll  t0, s1, 1
  sll  t0, t0, 2
  add  t0, s3, t0
  lw   a2, 0(t0)
  lw   a3, 4(t0)
  call chebyshev
  add  s4, gp, 48      <- Carga d[]
  sll  t0, s1, 2
  add  t0, s4, t0
  sw   a0, 0(t0)
  add  s1, s1, 1
  j    for
efor:
  j    .
```

# Ejercicio 18

## Funciones

- La función **abs** es de tipo hoja y no usará registros preservados.
  - No es necesario que salve el contexto.
- Recibe 1 argumento y retorna 1 resultado.
  - Por **a0** recibe dato cuyo valor absoluto hay que calcular.
  - Debe devolver el resultado también por **a0**.
  - Opera directamente con **a0**.

C/C++

```
int abs( int x )
{
  if( x < 0 )
    x = -x;
  return  x;
}
```

ASM

```
abs:
  bge a0, zero, else_abs
  neg a0, a0
else_abs:
  ret
```

- La función **chebyschev** es no-hoja y usará registros preservados.
  - Será necesario que salve la dirección de retorno y el contexto.
- Recibe 4 argumentos y retorna 1 resultado.
  - Las 2 coordenadas de cada punto las recibe por **a0..a3**.
  - Devuelve el resultado **a0**.

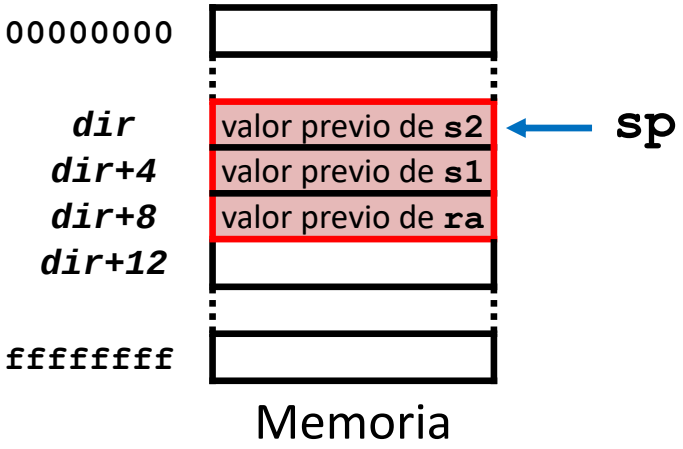
# Ejercicio 18

## Función **chebyschev** (variables locales en reg. preservados)

C/C++

```
int chebyshev(
  int x1, int y1,
  int x2, int y2 )
{
    int d1, d2;
    d1 = abs( x1-x2 );
    d2 = abs( y1-y2 );
    if( d2 > d1 )
        d1 = d2;
    return d1;
}
```

*x1 → a0, y1 → a1, x2 → a2 y2 → a3*
*d1 → s1, d2 → s2*




ASM

```
chebyshev:
  add   sp, sp, -3*4
  sw    ra, 8(sp)
  sw    s1, 4(sp)
  sw    s2, 0(sp)
  sub   a0, a0, a2
  call  abs
  mv    s1, a0
  sub   a0, a1, a3
  call  abs
  mv    s2, a0
if:
  ble   s2, s1, eif
  mv    s1, s2
eif:
  mv    a0, s1
  lw    ra, 8(sp)
  lw    s1, 4(sp)
  lw    s2, 0(sp)
  add   sp, sp, 3*4
  ret
```

*PROLOGO:*
*Apila contexto (`ra`, `s1` y `s2`)*
*(regla de la función invocada)*

`d1 = abs( x1-x2 )`

`d2 = abs( y1-y2 )`

¿ `d2` ≤ `d1` ?

`d1 = d2`

*Guarda en* `a0` *el resultado*

*EPILOGO:*
*Desapila contexto*

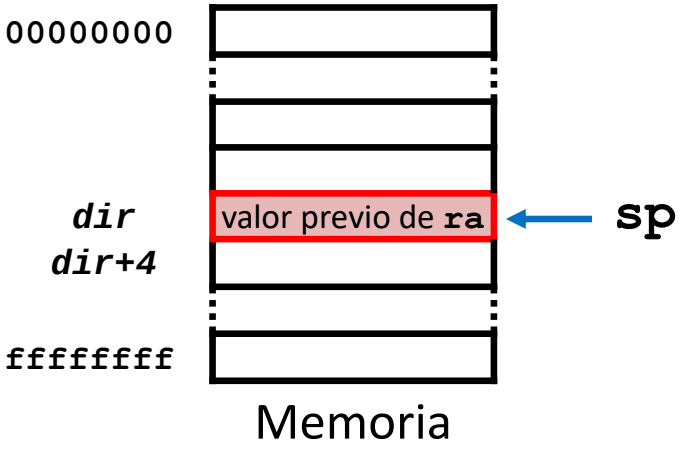
# Ejercicio 18

## Función **chebyschev** (variables locales en reg. temporales)

C/C++

```
int chebyshev(
  int x1, int y1,
  int x2, int y2 )
{
    int d1, d2;
    d1 = abs( x1-x2 );
    d2 = abs( y1-y2 );
    if( d2 > d1 )
        d1 = d2;
    return d1;
}
```

*x1 → a0, y1 → a1, x2 → a2 y2 → a3*
*d1 → t1, t2 → s2*




ASM

```
chebyshev:
  add   sp, sp, -4
  sw    ra, 0(sp)
  sub   a0, a0, a2
  call  abs
  mv    t1, a0
  add   sp, sp, -4
  sw    t1, 0(sp)
  sub   a0, a1, a3
  call  abs
  mv    t2, a0
  lw    t1, 0(sp)
  add   sp, sp, 4
if:
  ble   t2, t1, eif
  mv    t1, t2
eif:
  mv    a0, t1
  lw    ra, 0(sp)
  add   sp, sp, 4
  ret
```

*PROLOGO:*
*Apila contexto (`ra`)*
*(regla de la función invocada)*

`d1 = abs( x1-x2 )`

*Apila `t1` antes de la llamada*
*(regla de la función invocante)*

`d2 = abs( y1-y2 )`

*Desapila `t1` después de la llamada*

¿ `d2` ≤ `d1` ?

`d1 = d2`

*Guarda en `a0` el resultado*

*EPILOGO:*
*Desapila contexto*

# Ejercicio 18

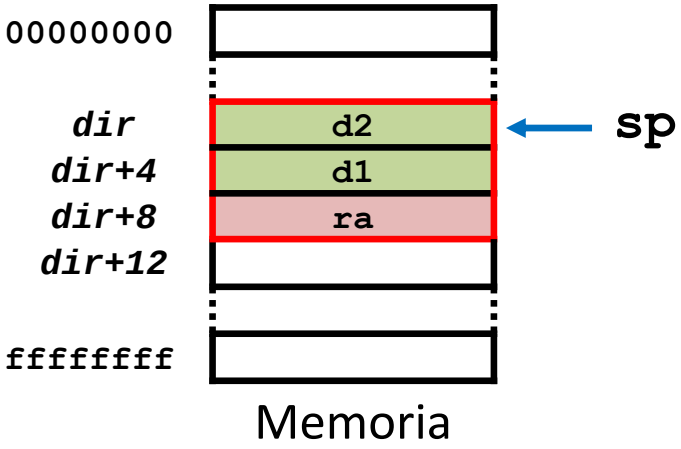
## Función **chebyschev** (variables locales en pila sin usar **fp**)

C/C++

```
int chebyshev(
  int x1, int y1,
  int x2, int y2 )
{
    int d1, d2;
    d1 = abs( x1-x2 );
    d2 = abs( y1-y2 );
    if( d2 > d1 )
        d1 = d2;
    return d1;
}
```

*x1 → a0, y1 → a1, x2 → a2 y2 → a3*

*d1 ≡ 4(sp), d2 ≡ 0(sp)*




ASM

```
chebyshev:
  add   sp, sp, -1*4
  sw    ra, 0(sp)
  add   sp, sp, -2*4
  sub   a0, a0, a2
  call  abs
  sw    a0, 4(sp)
  sub   a0, a1, a3
  call  abs
  sw    a0, 0(sp)
  lw    t1, 4(sp)
  lw    t2, 0(sp)
if:
  ble   t2, t1, eif
  mv    t1, t2
eif:
  mv    a0, t1
  add   sp, sp, 2*4
  lw    ra, 0(sp)
  add   sp, sp, 1*4
  ret
```

*PROLOGO:*
*Apila contexto (`ra`)*
*Reserva espacio para `d1` y `d2`*

`d1 = abs( x1-x2 )`

`d2 = abs( y1-y2 )`

¿ `d2` ≤ `d1` ?

`d1 = d2`

*Guarda en `a0` el resultado*

*EPILOGO:*
*Libera espacio de `d1` y `d2`*
*Desapila contexto*

# Ejercicio 18

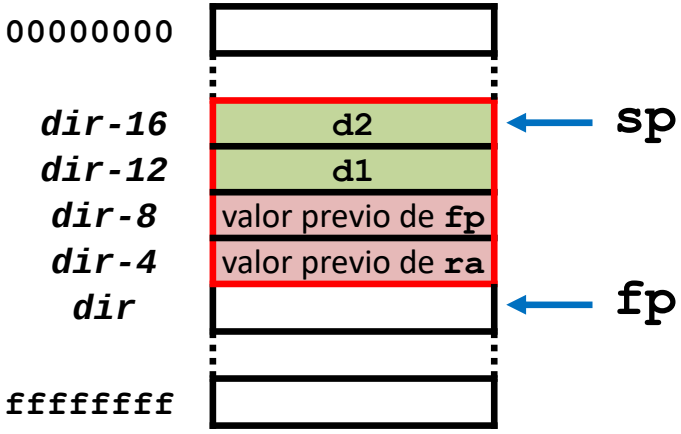
## Función **chebyschev** (variables locales en pila usando **fp**)

C/C++

```
int chebyshev(
  int x1, int y1,
  int x2, int y2 )
{
    int d1, d2;
    d1 = abs( x1-x2 );
    d2 = abs( y1-y2 );
    if( d2 > d1 )
        d1 = d2;
    return d1;
}
```

*x1 → a0, y1 → a1, x2 → a2 y2 → a3*

*d1 ≡ -12(fp), d2 ≡ -16(fp)*




ASM

```
chebyshev:
  add   sp, sp, -2*4
  sw    ra, 4(sp)
  sw    fp, 0(sp)
  add   fp, sp, 2*4
  add   sp, sp, -2*4
  sub   a0, a0, a2
  call  abs
  sw    a0, -12(fp)
  sub   a0, a1, a3
  call  abs
  sw    a0, -16(fp)
  lw    t1, -12(fp)
  lw    t2, -16(fp)
if:
  ble   t2, t1, eif
  mv    t1, t2
eif:
  mv    a0, s1
  add   sp, sp, 2*4
  lw    ra, 4(sp)
  lw    fp, 0(sp)
  add   sp, sp, 2*4
  ret
```

*PROLOGO:*
*Apila contexto (`ra` y `fp`)*
*Actualiza `fp`*
*Reserva espacio para `d1` y `d2`*

`d1 = abs( x1-x2 )`

`d2 = abs( y1-y2 )`

¿ `d2` ≤ `d1` ?

`d1 = d2`

*Guarda en `a0` el resultado*

*EPILOGO:*
*Libera espacio de `d1` y `d2`*
*Desapila contexto*

# Acerca de *Creative Commons*

- Licencia CC (Creative Commons)


  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/