Module 3 – Problems (Bonus)

Programming in assembly

Introduction to computers II

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

Problem 18

Pseudo-code

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);
}
```

Problem 18

Global variables

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];
```

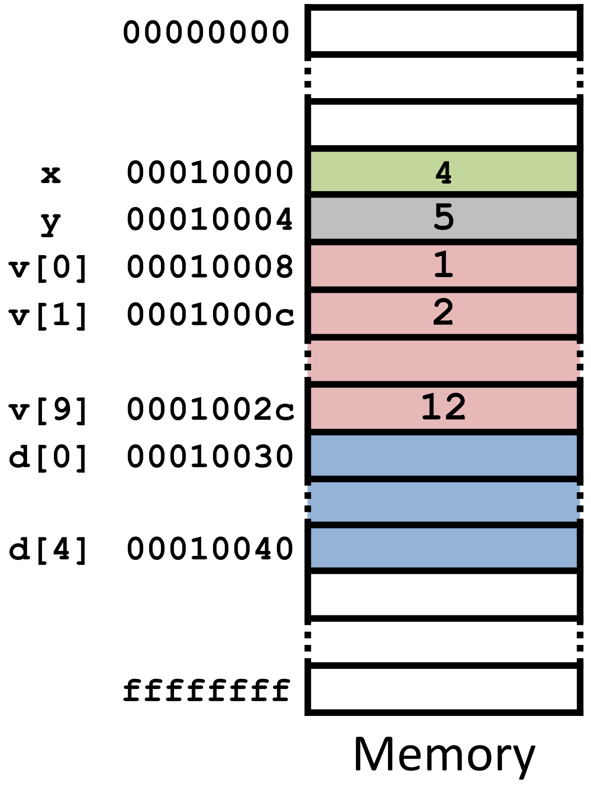
ASM

```
.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
```



Problem 18

Main program (using labels)

C/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

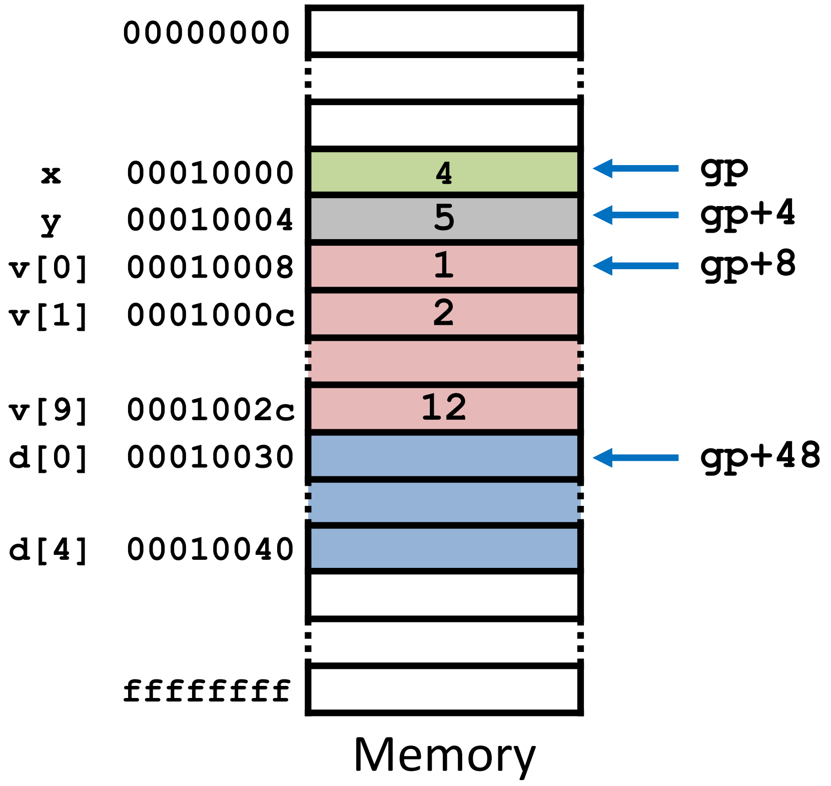
C/C++

```
main:
   la     sp, _stack      ← Stack initialization
   mv     s1, zero
   li     s2, N
for:
   bge    s1, s2, efor    ┐
   la     t0, x           │
   lw     a0, 0(t0)       │
   la     t0, y           │ Pass parameters
   lw     a1, 0(t0)       │
   la     s3, v           │
   sll    t0, s1, 1       │ ← Calculate i*2
   sll    t0, t0, 2       │ ← Calculate the offset
   add    t0, s3, t0      │ ← Add base and offset
   lw     a2, 0(t0)       │ ← Load v[2*i]
   lw     a3, 4(t0)       ┘ ← Load v[2*i+1]
   call   chebyshev
   la     s4, d           ┐
   sll    t0, s1, 2       │ Store result
   add    t0, s4, t0      │
   sw     a0, 0(t0)       ┘ ← Store in d[i]
   add    s1, s1, 1
   j      for
efor:
   j      .
```

Problem 18

Main program (using gp)

i → s1, N → s2, v[] → s3, d[] → s4



C/C++

```
main:
   la     sp, _stack        Stack initialization
   la     gp, x             Initialize gp with the first address of data
   mv     s1, zero
   li     s2, N
for:
   bge    s1, s2, efor
   lw     a0, 0(gp)         Load x
   lw     a1, 4(gp)         Load y
   add    s3, gp, 8         Load 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        Load d[]
   sll    t0, s1, 2
   add    t0, s4, t0
   sw     a0, 0(t0)
   add    s1, s1, 1
   j      for
efor:
   j      .
```

Problem 18

Functions

- The **abs** function is a leaf function and will not used preserved registers.
  - It does not have to save the context.
- It receives 1 argument and returns 1 result.
  - It receives in **a0** the data whose absolute value has to be calculated.
  - The result has to be returned in **a0**.
  - It operates with **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
```

- The **chebyshev** function is a non-leaf function and will use preserved registers.
  - It must save the context and the return address.
- It receives 4 arguments and returns 1 result.
  - The 2 coordinates of each point will be passed through **a0..a3**.
  - It returns the result through **a0**.

Problem 18

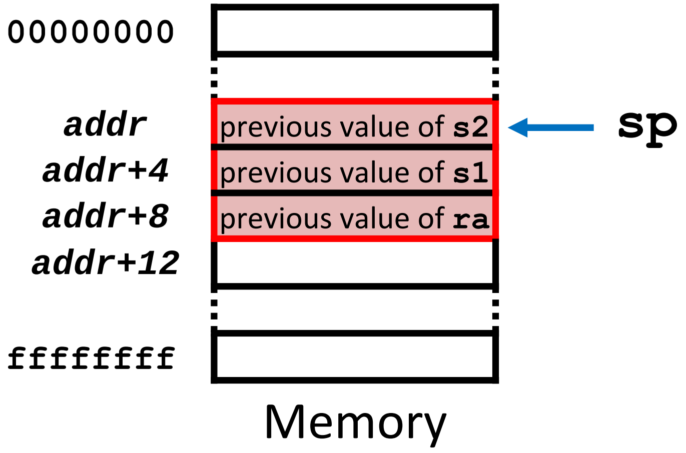
chebyshev function (local variables in preserved reg.)

C/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

```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
```

PROLOGUE: *Save context (`ra`, `s1` and `s2`)*

`d1 = abs( x1-x2 )`

`d2 = abs( y1-y2 )`

`d2 ≤ d1 ?`

`d1 = d2`

Store the result in `a0`

EPILOGUE: *Restore context*

Problem 18

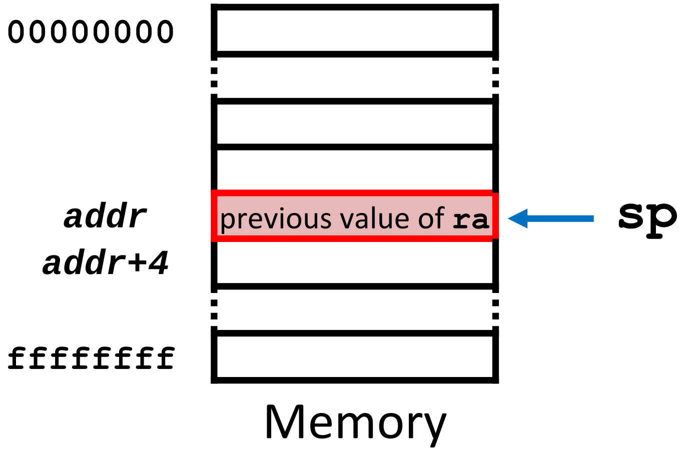
chebyshev function (local variables in temporary reg.)

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      PROLOGUE:
   sw    ra, 0(sp)       Save context (ra, s1 and s2)
   sub   a0, a0, a2
   call  abs             d1 = abs( x1-x2 )
   mv    t1, a0
   add   sp, sp, -4
   sw    t1, 0(sp)       Push t1 before the call
   sub   a0, a1, a3
   call  abs             d2 = abs( y1-y2 )
   mv    t2, a0
   lw    t1, 0(sp)
   add   sp, sp, 4       Pop t1 after the call
if:
   ble   t2, t1, eif     d2 ≤ d1 ?
   mv    t1, t2          d1 = d2
eif:
   mv    a0, t1          Store the result in a0
   lw    ra, 0(sp)
   add   sp, sp, 4       EPILOGUE:
   ret                   Restore context
```

Problem 18

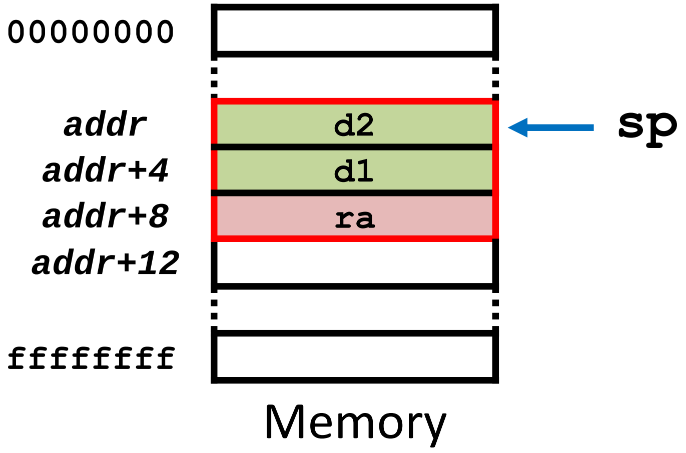
chebyshev function (local variables in stack without using fp)

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      PROLOGUE:
   sw    ra, 0(sp)         Save context (ra)
   add   sp, sp, -2*4      Reserve space for d1 and d2
   sub   a0, a0, a2
   call  abs               d1 = abs( x1-x2 )
   sw    a0, 4(sp)
   sub   a0, a1, a3
   call  abs               d2 = abs( y1-y2 )
   sw    a0, 0(sp)
   lw    t1, 4(sp)
   lw    t2, 0(sp)
if:                        d2 ≤ d1 ?
   ble   t2, t1, eif
   mv    t1, t2            d1 = d2
eif:
   mv    a0, t1            Store the result in a0
   add   sp, sp, 2*4       EPILOGUE:
   lw    ra, 0(sp)         Free space of d1 and d2
   add   sp, sp, 1*4       Restore context
   ret
```

Problem 18

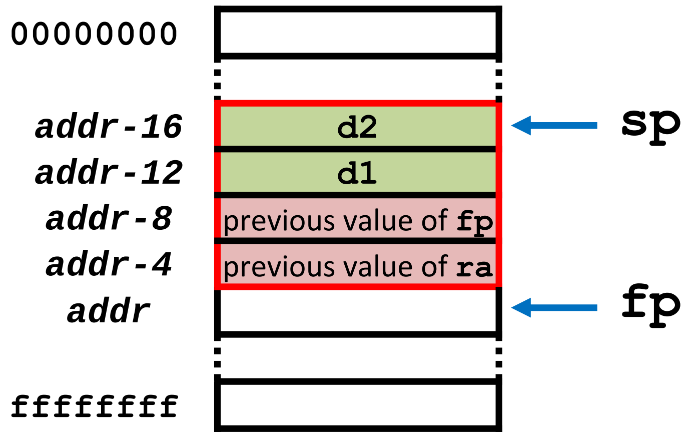
chebyshev function (local variables in stack using fp)

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
```

PROLOGUE:
Save context (`ra` and `fp`)
Update `fp`
Reserve space for `d1` and `d2`

`d1 = abs( x1-x2 )`

`d2 = abs( y1-y2 )`

`d2 ≤ d1 ?`

`d1 = d2`

Store the result in `a0`

EPILOGUE:
Free space of `d1` and `d2`
Restore context

About *Creative Commons*