# Paso de C a ensamblador


Facultad de Informática

# Índice de contenidos

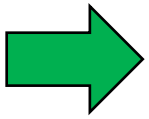
- Construcciones de control básicas
- Llamadas a funciones y paso de parámetros
- Marco de pila
- Variables locales y globales

# Índice de contenidos

# Ejecución condicional

- Uso de una secuencia de instrucciones condicionales
  - Suponiendo a guardado en R0, b guardado en R1 y count en R2

```
if (a==b) {
  count+=1;
}
else {
  count-=1;
}
```

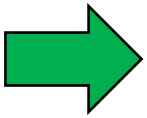


```
CMP R0,R1
ADDEQ R2,R2,#1
SUBNE R2,R2,#1
```

# Ejecución condicional

- Fijar los flags, y después usar varios códigos de condición

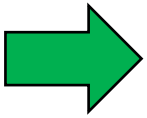
```
if (a==0)
  x=0;
if (a>0)
  x=1;
```



```
CMP R0,#0
MOVEQ R1,#0
MOVGT R1,#1
```

- Uso de instrucciones de comparación condicionales

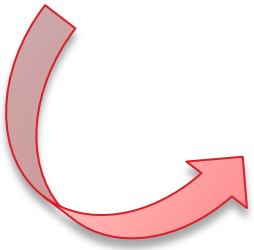
```
if (a==4 || a==10)
  x=0;
```



```
CMP R0,#4
CMPNE R0,#10
MOVEQ R1,#0
```

# if-then-else general

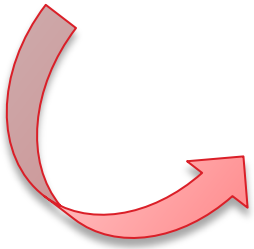
```
if(cond) {
  caso cierto;
} else {
  caso falso;
}
```



```
IF:
   B<NOT cond> CASO_FALSO
   ; ejecución caso cierto
   ...
   B REAGRUPAR
CASO_FALSO:
    ; ejecución caso falso
    ...
REAGRUPAR:
    ; inst. flujo reagrupado
```

# Bucle while

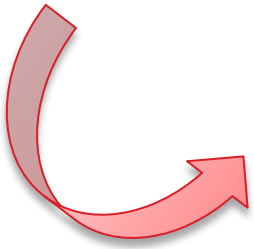
```
while (cond) {
  ejec.caso cierto;
}
```



```
BUCLE:
            B<NOT cond> casoFalso
            ; ejecución instruc. Condición cierta
            B bucle
CASO_FALSO:
            ; inst. tras bucle
```

# Bucle for

```
count=0;
for (i=10;i>0;i--)
  count+=1;
```



```
            MOV R2,#0
            MOV R0,#10
BUCLE_FOR:
            ADD R2,R2,#1
            SUBS R0,R0,#1
            BNE BUCLE_FOR
            ; instrucciones tras bucle
```

# Índice de contenidos

# Llamadas a funciones y paso de parámetros

- Llamada a una subrutina
  - Almacenar la dirección de retorno en r14 (LR)
  - Realizar un salto a la dirección de comienzo
- Dos formas:
  - Con instrucción Branch and Link (BL)
  - Con una secuencia de instrucciones
- Retorno:
  - Copiar en PC el valor almacenado en LR

# Llamadas a funciones y paso de parámetros

- Llamada con la instrucción BL (1)

```
BL Etiqueta ; Llamada, LR = PC - 4 = . + 4
…
Etiqueta: … ; Dirección de comienzo
          … ; Instrucciones de la rutina
          MOV PC,LR ; Retorno
```

- Limitaciones (*):
  - La rutina debe estar en la misma sección de código que la llamada
  - El ensamblador debe poder resolver la dirección

# Llamadas a funciones y paso de parámetros

- Llamada con una secuencia de instrucciones (2)

```
MOV LR,PC         ; LR = PC = . + 8
LDR PC,=Etiqueta  ; Llamada
…
Etiqueta: … ; Dirección de comienzo
          … ; Instrucciones de la rutina
          MOV PC,LR ; Retorno
```

- Rutina y llamada pueden estar en secciones distintas
  - El enlazador resuelve la dirección
- La dirección puede estar en un registro y se puede usar un MOV para saltar

# Llamadas a funciones y paso de parámetros

- Se sigue un estándar
  - AAPCS: ARM Architecture Procedure Call Standard
- Define la forma de generar código para mezclar objetos creados de forma distinta
  - Por ejemplo mezclar código ensamblador con código C
- El AAPCS regula:
  - La forma de pasar parámetros a las rutinas
  - Los registros que deben preservarse en la rutina
  - La estructura de las rutinas
- La estructura del marco de pila

# Llamadas a funciones y paso de parámetros

- Reglas AAPCS de paso de parámetros enteros:
  - Los cuatro primeros parámetros se pasan por registro
    - R0, R1, R2 y R3 (A1, A2, A3 y A4)
  - El resto se pasan por pila en sentido inverso
    - De este modo quedan ordenados
- Nota: Comportamiento gcc con -O0
  - A pesar de pasar los parámetros por registros se escriben en memoria (redundancia)

# Llamadas a funciones y paso de parámetros

- Ejemplo

```
void funB(int Param1, int Param2, int Param3, int Param4,
          int Param5, int Param6, int Param7)
{
  cuerpo
}
```

```
LDR R0,=Param5
STR R0,[SP]
LDR R0,=Param6
STR R0,[SP,#4]
LDR R0,=Param7
STR R0,[SP,#8]
...
```

# Índice de contenidos

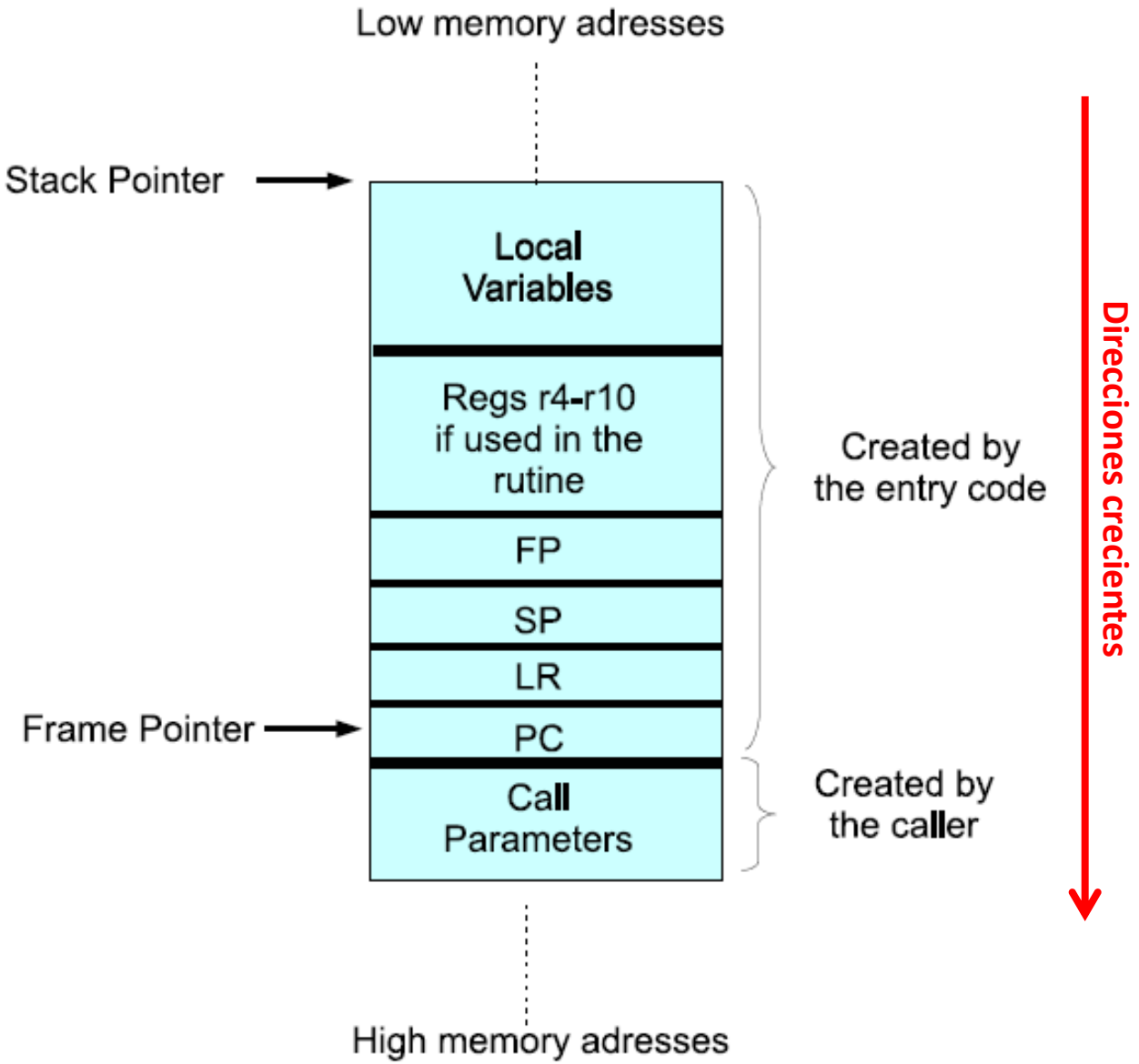
# Llamadas a funciones y paso de parámetros

- Reglas AAPCS para el valor de retorno de enteros:
  - Si es una palabra, se devuelve por r0
  - Si son 2-4 palabras, se devuelve por r0-r3
  - Si es mayor, se devuelve en memoria a través de una dirección pasada como parámetro
- En cuanto a los registros:
  - R0-R3 (A1-A4): Para pasar argumentos, devolver resultado y guardar datos temporales dentro de la rutina
  - R4-R11 (V1-V8): Para almacenar variables dentro de la rutina
  - R12 (IP): Registro auxiliar
  - R13(SP), R14(LR), R15 (PC): Registros de uso especial
  - R11 (FP) también es especial en algunas variantes de AAPCS
  - Es obligatorio preservar r4-r11 y r13

# Marco de pila

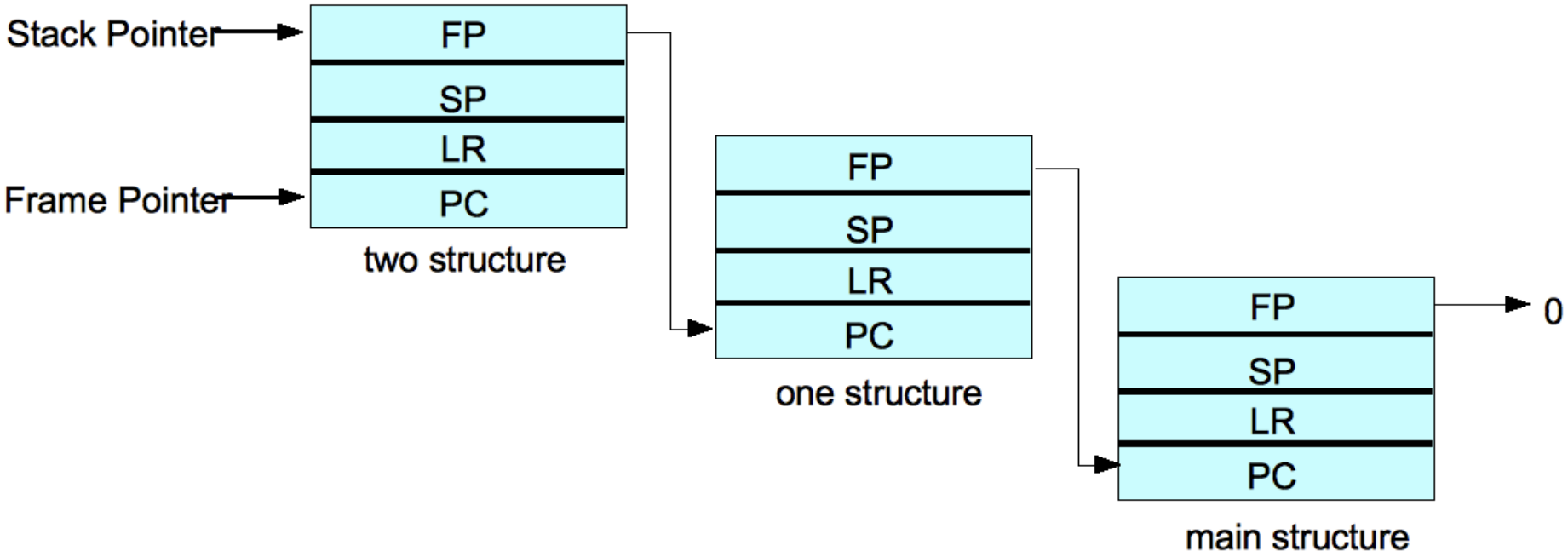
- AAPCS asume una pila Full Descending
- Cada rutina crea un marco en la pila con:
  - Los registros que debe preservar
  - La dirección de retorno
  - Las variables locales de la rutina
- SP apunta a la cima del marco
- FP apunta a la base del marco
  - Las variables locales son accedidas a través del FP

# Marco de pila



# Marco de pila

- Ejemplo



# Marco de pila

- Estructura de una rutina

Código de entrada (prólogo) → Construye el marco
Cuerpo de la rutina
Código de salida (epílogo) → Destruye el marco y hace el retorno

```
Estructura de una rutina
Prólogo:
MOV   IP,SP
STMDB SP!,{R4-R10,FP,IP,LR,PC}
SUB   FP,IP,#4
SUB   SP,#EspacioParaVariablesLocales
Epílogo:
LDMDB FP, {R4-R10,FP,SP,PC}
```

# Índice de contenidos

# Variables locales y globales

- Variables globales
  - Almacenadas en secciones .data o .bss
  - Persisten en memoria durante todo el programa
- Variables locales
  - Almacenadas en el marco de pila de la rutina
  - Activas sólo en el cuerpo de la rutina