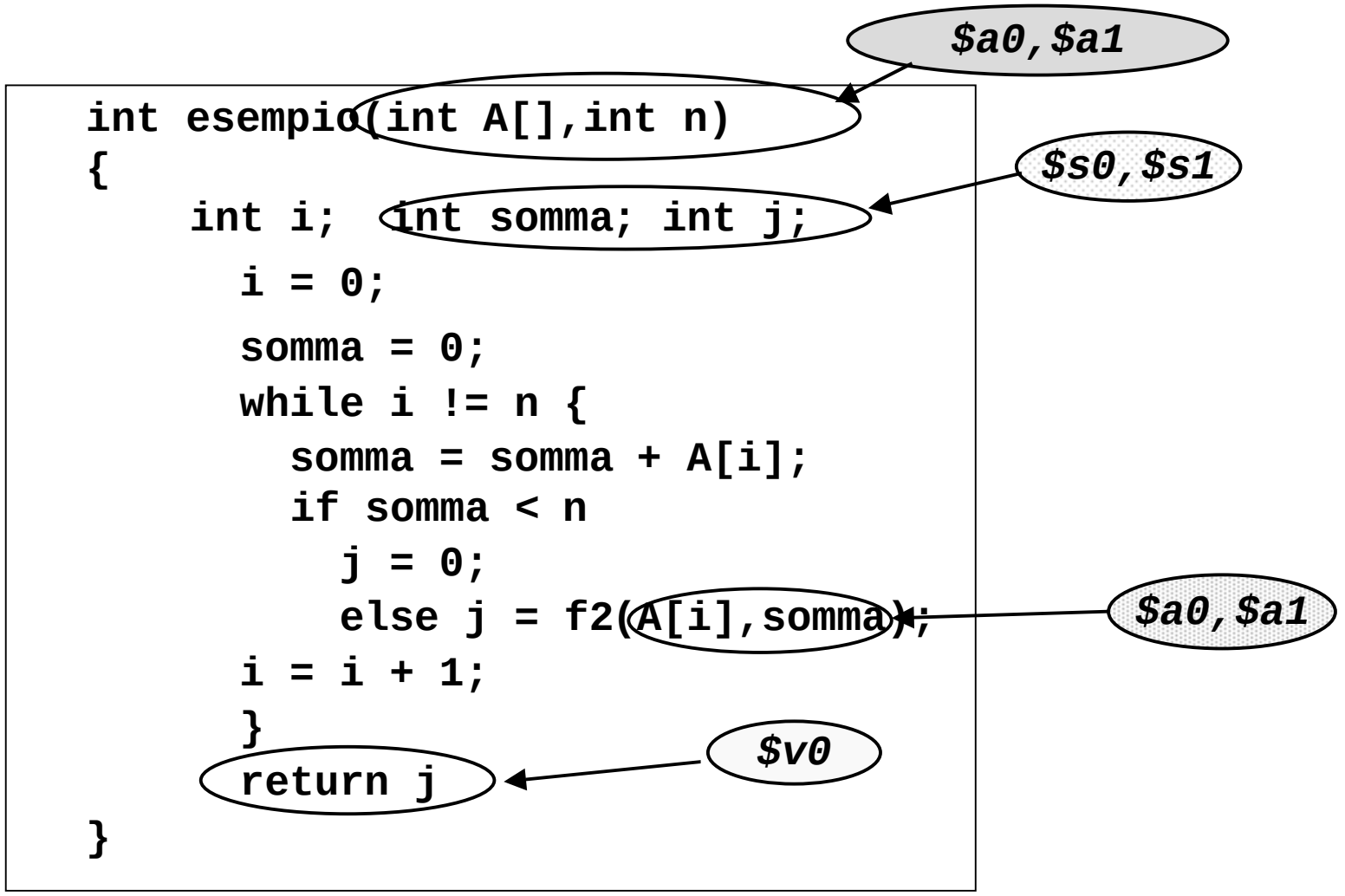
Esercizio 1

Compilare la seguente function



Esempio sul valore soglia (2)

Var	Registro
i	$s1
n	$s2
base A	$s0

```
i = 0;
while i < n
{
   <elabora A[i]>;
   i = i + 1;
}
```

In generale : `valore_soglia = base + 4*n`

- versione senza contatore

```
inizializzo puntatore;
calcolo valore soglia;
while punt < soglia
{
   <elabora variabile puntata>
   avanza di 4 byte:
   }
```

```
      add $t0,$s0,$zero # $t0=base
      add $t1,$s2,$s2  # 2n
      add $t1,$t1,$t1  # 4n
      add $t1,$t1,$s0  # 4n+base
loop: beq $t0,$t1,exit # $t0=soglia?
      .. ..
      addi $t0,$t0,4   # avanza
      j loop
exit: .. ..
```

Convenzioni

- **`$t0` scandisce `A`**
- **`$t1` è la soglia per `$t0`**
- **in `$t2` carico `A[i]`**
- ***Salviamo nello stack*:**
  - **`$s0`, `$s1`, `$ra` perché è chiamato**
  - **`$a1`, `$t0`, `$t1` perchè è chiamante**
- ***Nota:***
  - **`$a0` è usato solo al di fuori del ciclo (per inizializzare il puntatore e calcolare la soglia), quindi non è indispensabile salvarlo**
  - **`$t2` cambierà valore dopo la chiamata, quindi non è indispensabile salvarlo**
- **in tutto devo salvare 6 elementi = 4*6 byte = 24 byte**

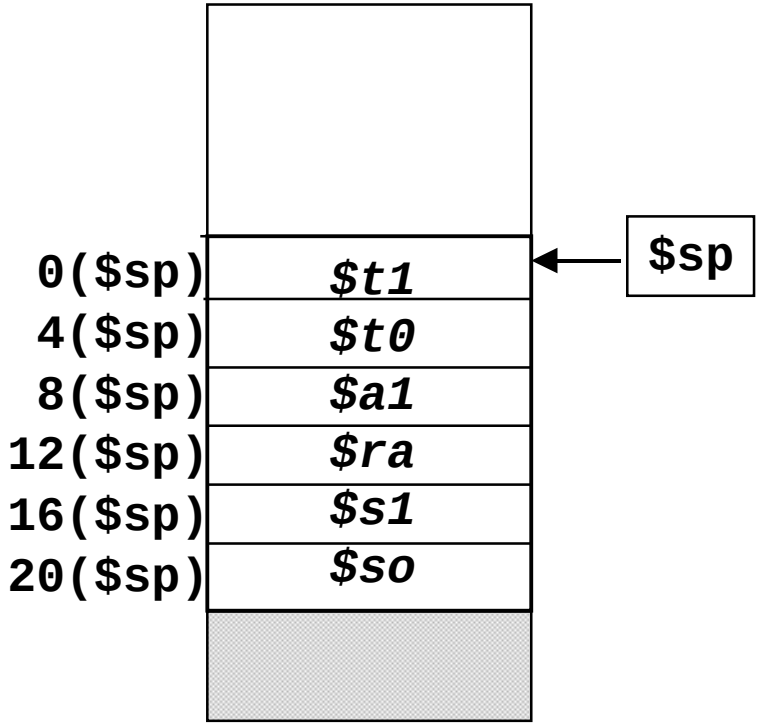
Progetto il frame

Bisogna salvare: `$s0,$s1,$ra,$a1,$t0,$t1` (6 elementi)

1. alloco spazio per i 6 elementi

2. decido in che ordine li salvo

2. determino l'indirizzo di ogni parola del frame



Soluzione dell'esercizio 1 (1)

- ***Alloc./dealloc. frame***
- ***Salvataggio reg. "salvati" e $ra***

```
esempio:addi $sp,$sp,-24
        sw   $s0,20($sp)
        sw   $s1,16($sp)
        sw   $ra,12($sp)
      <corpo subroutine>
        lw   $ra,12($sp)
        lw   $s1,16($sp)
        lw   $s0,20($sp)
        addi $sp,$sp,24
        jr   $ra
```

- ***Corpo subroutine (1)***
  - **gestione scansione**

```
        add $t0,$a0,$zero # $t0 punta a A
        add $t1,$a1,$a1   # 2n
        add $t1,$t1,$t1   # 4n
        add $t1,$t1,$a0   # 4n + base
loop:   beq $t0,$t1,exit  # $t0 = soglia?
        <corpo del ciclo>
        addi $t0,$t0,4    # avanza in A
        j loop
exit:   add $v0,$s1,$zero # restituisco j
        É         # si ricollega a lw ...
```

oppure:

```
sll $t1,$a1,2
```

Soluzione dell'esercizio 1 (2)

- ***Corpo subroutine (2)***
  - **corpo del ciclo**

```
        lw   $t2,0($t0)     # prelevo A[i] dalla memoria

        add  $s0,$s0,$t2    # somma = somma + A[i]
        slt  $t3,$s0,$a1    # $t3 = 1 se somma < n
        bne  $t3,$zero,SI   # se $t3 = 1 (somma < n) vai
                            # all'istruzione con etichetta SI
        <gestione della chiamata>
        add  $s1,$v0,$zero  # prelevo valore restituito
        j    fuori          # esci dalla if-then-else
  SI: add  $s1,$zero,$zero# j = 0
fuori:     É                # si ricollega con addi ...
```

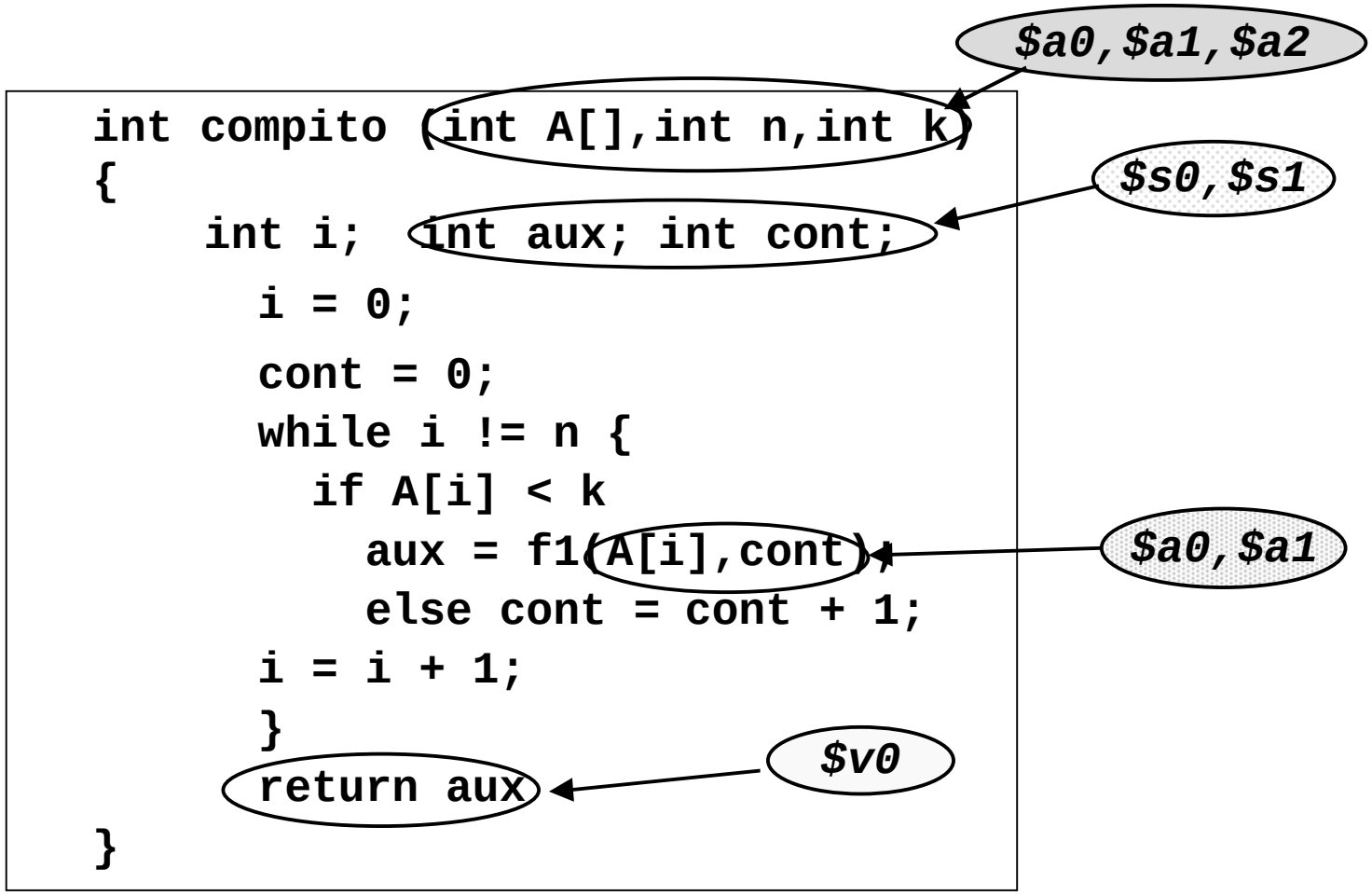
Soluzione dell'esercizio 1 (3)

- ***Gestione della chiamata***

```
sw   $a1,8($sp)
sw   $t0,4($sp)
sw   $t1,0($sp)
add  $a0,$t2,$zero  # passaggio parametri
add  $a1,$s0,$zero  # passaggio parametri
jal  f2
lw   $t1,0($sp)
lw   $t0,4($sp
lw   $a1,8($sp)
```

Esercizio 2

Compilare la seguente function



Scansione con valore soglia

Var	Registro
i	$s1
n	$s2
base A	$s0

```
i = 0;
while i < n
{
   <elabora A[i]>;
   i = i + 1;
}
```

In generale: `valore_soglia = base + 4*n`

```
inizializzo puntatore;
calcolo valore soglia;
while punt < soglia
{
   <elabora variabile puntata>
   avanza di 4 byte:
   }
```

```
      add $t0,$s0,$zero # $t0=base
      add $t1,$s2,$s2  # 2n
      add $t1,$t1,$t1  # 4n
      add $t1,$t1,$s0  # 4n+base
loop: beq $t0,$t1,exit # $t0=soglia?
      .. ..
      addi $t0,$t0,4   # avanza
      j loop
exit: .. ..
```

Convenzioni

- **`$t0` scandisce `A`**
- **`$t1` è la soglia per `$t0`**
- **in `$t2` carico `A[i]`**
- ***Salviamo nello stack*:**
  - **`$s0`, `$s1`, `$ra` perché è chiamato**
  - **`$a2`, `$t0`, `$t1` perchè è chiamante**
- ***Nota:***
  - **`$a0`,`$a1` sono usati solo al di fuori del ciclo (per inizializzare il puntatore e calcolare la soglia), quindi non è indispensabile salvarli**
  - **`$t2` cambierà valore dopo la chiamata, quindi non è indispensabile salvarlo**
- **in tutto devo salvare 6 elementi = 4*6 byte = 24 byte**

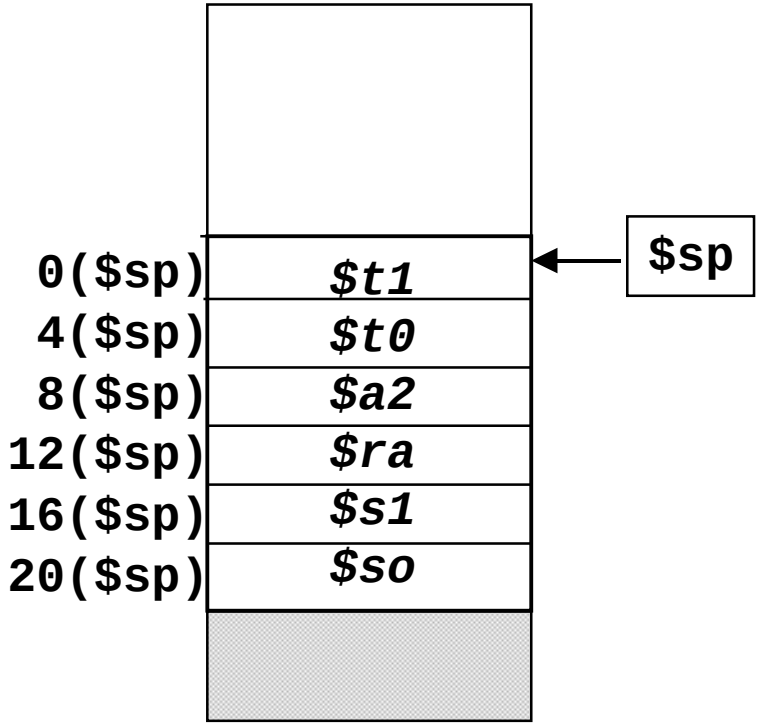
Progetto il frame

Bisogna salvare: `$s0,$s1,$ra,$a2,$t0,$t1` (6 elementi)

1. alloco spazio per i 6 elementi

2. decido in che ordine li salvo

2. determino l'indirizzo di ogni parola del frame



Soluzione dell'esercizio 2 (1)

- ***Alloc./dealloc. frame***
- ***Salvataggio reg. "salvati" e $ra***

```
esempio:addi $sp,$sp,-24
        sw   $s0,20($sp)
        sw   $s1,16($sp)
        sw   $ra,12($sp)
      <corpo subroutine>
        lw   $ra,12($sp)
        lw   $s1,16($sp)
        lw   $s0,20($sp)
        addi $sp,$sp,24
        jr   $ra
```

- ***Corpo subroutine (1)***
  - **gestione scansione**

```
        add $t0,$a0,$zero # $t0 punta a A
        add $t1,$a1,$a1   # 2n
        add $t1,$t1,$t1   # 4n
        add $t1,$t1,$a0   # 4n + base
loop:   beq $t0,$t1,exit  # $t0 = soglia?
        <corpo del ciclo>
        addi $t0,$t0,4    # avanza in A
        j loop
exit:   add $v0,$s0,$zero #restituisco aux
        É         # si ricollega a lw ...
```

oppure:

```
sll $t1,$a1,2
```

Soluzione dell'esercizio 2 (2)

- *Corpo subroutine (2)*
  - **corpo del ciclo**

```
        lw   $t2,0($t0)     # prelevo A[i] dalla memoria
        slt  $t3,$t2,$a2    # $t3 = 1 se A[i] < k
        bne  $t3,$zero,SI   # se $t3 = 1 (A[i] < k) vai
                            # all'istruzione con etichetta SI
        addi $s1,$s1,1      # cont + 1, eseguita solo se A[i] ≥ k
        j    fuori          # esci dalla if-then-else
  SI: <gestione della chiamata
        add  $s0,$v0,$zero  # prelevo valore restituito
fuori:  É                   # si ricollega con addi ...
```

Soluzione dell'esercizio 2 (3)

- ***Gestione della chiamata***

```
sw   $a2,8($sp)
sw   $t0,4($sp)
sw   $t1,0($sp)
add  $a0,$t2,$zero  # passaggio parametri
add  $a1,$s1,$zero  # passaggio parametri
jal  f1
lw   $t1,0($sp)
lw   $t0,4($sp
lw   $a2,8($sp)
```