TRADUZIONE DI FUNZIONI E COSTRUTTI PRINCIPALI

ESERCIZI DI ESEMPIO

NOTA: tutti gli esercizi includono anche la chiamata alla funzione C "exit(0)" alla fine della funzione main per essere eseguiti correttamente in RARS; tale chiamata tuttavia non è necessaria nello svolgimento degli esercizi

1) SUM

```
long int sum(long int a, long int b){
        long int d;
        d = a + b;
        return d;
}


void main(){
        long int a, b, c;
        a = 5;
        b = 3;
        c = sum(a, b);
        return;
}
```

Soluzione

```
.text
.globl MAIN
MAIN:
      # $s0 = a, $s1 = b, $s2 = c
      li  s0, 5
      li  s1, 3
      mv a2, s0
      mv a3, s1
      jal SUM   #jal ra, SUM
      mv s2, a0 #add s2, zero, a0
      mv a2, s2 #add s2, zero, s2
      # per terminare l'esecuzione tramite exit
      li a0, 0
      li a7, 93
      ecall
SUM:
      addi sp, sp, -8
      sd  s0, 0(sp)
      add  s0, a2, a3
      mv   a0, s0
      ld   s0, 0(sp)
      addi sp, sp, 8
      jr   ra     #jalr zero, 0(ra)
```

2) PUNTATORE

```
long int v[10];
long int *p;

void foo(long int n){
        long int i;
        p = v;
        for(i = 0; i < 10; i++){
                *p = *p + n;
                p++;
        }
}


void main(){
        foo(5);
}
```

Soluzione

```
.data
V: .space 80 # array di 10 elementi
P: .dword 0 # puntatore inizializzato a NULL

.text
.globl MAIN

MAIN:
      li  a2, 5
      jal  FOO
      # per terminare l'esecuzione tramite exit
      li a0, 0
      li a7, 93
      ecall
FOO:
      addi  sp, sp, -8
      sd    s0, 0(sp)   # backup di $s0

      la    t0, V
      la    t1, P
      sd    t0, 0(t1)   # carica contenuto di $t0 in P


      li    s0, 0       # inizializza i

      li    t2, 10
FOR:  bge   s0, t2, ENDFOR
      ld    t0, P       # carica contenuto di P in $t0
      ld    t1, 0(t0)   # carica *p
      add   t1, t1, a2  # somma n

      sd    t1, 0(t0)   # salva *p
      ld    t0, P       # ri-carica contenuto di P in $t0 (non
necessario)
      addi  t0, t0, 8   # incrementa contenuto di P

      la    t1, P
      sd    t0, 0(t1)   # salva contenuto di P

      addi  s0, s0, 1
      j FOR

ENDFOR:
      ld   s0, 0(sp)
      addi sp, sp, 8
      jr   ra
```

3) POTENZA

```
long int potenza(long int n, long int exp){

        if(exp == 0){
                return 1; }
        else{
                return n * potenza(n, exp -1);
        }
}


void main(){

        potenza(2, 5);

}
```

Soluzione

```
.text
.global MAIN


MAIN:
      li a2, 2
      li a3, 5
      # per stampare a schermo il contenuto di a0
      jal POTENZA
      mv a0, a0,
      li a7, 1
      ecall
      # per terminare l'esecuzione tramite exit
      li a0, 0
      li a7, 93
      ecall
POTENZA:
      addi sp, sp, -8
      sd   ra, 0(sp)
      bnez a3, ELSE
      li a0, 1
      j EXIT
ELSE:
      addi sp, sp, -8
      sd   a2, 0(sp)
      addi a3, a3, -1
      jal POTENZA
      ld a2, 0(sp)
      addi sp, sp, 8
      mul a0, a0, a2
EXIT:
      lw ra, 0(sp)
      addi sp, sp, 8
      jr ra
```

4) BIG_FUNC – senza e con $fp

```
long int sums(long int n, long int m) {
        return n * m + 17;
 }
long int big_fun(long int a, long int b, long int c, long int d, long int e) {
        return c + sums(d, g) + a * b + f + e;

}
void main() {
        int a = 1, b = 2, c = 3, d = 4, e = 5, f = 6, g = 7, res;
        res = big_fun(a, b, c, d, e, f, g);
 }
```

Soluzione - **Senza $fp**

```
.text
.globl MAIN

MAIN:
        # s0 = a, s1 = b, s2 = c, s3 = d, s4 = e, s5 = f, s6 = g, s7 = risultato
        li      s0, 1
        li      s1, 2
        li      s2, 3
        li      s3, 4
        li      s4, 5
        li      s5, 6
        li      s6, 7

        # inizializza registry parametri BIG_FUN
        mv      a2, s0
        mv      a3, s1
        mv      a4, s2
        mv      a5, s3
        mv      a6, s4
        mv      a7, s5

        # passa quinto parametron su stack
        addi    sp, sp, -8
        sd      s6, 0(sp)

        jal     BIG_FUN         # chiama BIG_FUN

        addi    sp, sp, 8       # libera spazio sullo stack

        mv      s7, a0
        # per terminare l'esecuzione tramite exit
        li a0, 0
        li a7, 93
        ecall


BIG_FUN:
        # backup PC ritorno e registry parametri
        addi    sp, sp, -8
        sd      ra, 0(sp)

        # backup a2 - a7
        addi    sp, sp, -48
        sw      a2, 40(sp)
        sw      a3, 32(sp)
```

```
        sw      a4, 24(sp)
        sw      a5, 16(sp)
        sw      a6, 8(sp)
        sw      a7, 0(sp)

        # prepara registry a per chiamata a SUMS
        mv      a2, a5
        ld      a3, 56(sp)      # leggi il parametro “g” dallo stack
        jal     SUMS            #chiama SUMS

        # ripristina a2 - a7
        lw      a2, 40(sp)
        lw      a3, 32(sp)
        lw      a4, 24(sp)
        lw      a5, 16(sp)
        lw      a6, 8(sp)
        lw      a7, 0(sp)
        addi    sp, sp, 48

        # procedi con l’espressione
        add     t0, a0, a4
        mul     t1, a2, a3
        add    t2, t1, t0
        add    t3, t2, a7
        add     a0, t3, a6

        ld      ra, 0(sp)       # ripristina ra
        addi    sp, sp, 8
        jr      ra

SUMS:
        # calcola espressione
        mul     t0, a0, a1
        addi    a0, t0, 17
        jr      ra
```

Soluzione - **Con $fp**

```
.text
.globl MAIN

MAIN:
        # s0 = a, s1 = b, s2 = c, s3 = d, s4 = e, s5 = f, s6 = g, s7 = res
        li      s0, 1
        li      s1, 2
        li      s2, 3
        li      s3, 4
        li      s4, 5
        li      s5, 6
        li    s6, 7

        # inizializza registry parametri BIG_FUN
        mv      a2, s0
        mv      a3, s1
        mv      a4, s2
        mv      a5, s3
        mv     a6, s4
        mv      a7, s5

        # passa quinto parametron su stack
        addi    sp, sp, -8
        sd      s6, 0(sp)

        jal     BIG_FUN         # chiama BIG_FUN

        addi    sp, sp, 8       # libera spazio sullo stack

        mv      s7, a0
        # per terminare l'esecuzione tramite exit
        li a0, 0
        li a7, 93
        ecall


BIG_FUN:
        # backup PC ritorno e registry parametri
        addi    sp, sp, -16
        sd      fp, 8(sp)       # salva fp precedente
        addi    fp, sp, 8       # fp punta all'inizio del record di attivazione di BIG_FUN
        sw      ra, 0(sp)

        # backup a2 - a7
        addi    sp, sp, -48
```

```
        sw      a2, 40(sp)
        sw      a3, 32(sp)
        sw      a4, 24(sp)
        sw      a5, 16(sp)
        sw      a6, 8(sp)
        sw      a7, 0(sp)

        # prepara registry a per chiamata a SUMS
        mv      a2, a5
        ld      a3, 8(fp)       # leggi il parametro "g" dallo stack
        jal     SUMS            #chiama SUMS

        # ripristina a2 - a7
        lw      a2, 40(sp)
        lw      a3, 32(sp)
        lw      a4, 24(sp)
        lw      a5, 16(sp)
        lw      a6, 8(sp)
        lw      a7, 0(sp)
        addi    sp, sp, 48

        # procedi con l'espressione
        add     t0, a0, a4
        mul     t1, a2, a3
        add    t2, t1, t0
        add    t3, t2, a7
        add     a0, t3, a6

        ld      ra, -8(fp)      # ripristina ra dallo stack tramite fp
        addi    sp, fp, 8       # ripristina valore iniziale di sp usando fp
        ld      fp, 0(fp)       # ripristina fp dallo stack tramite fp

        jr      ra

SUMS:
        addi    sp, sp, 8       # alloca spazio sullo stack
        sd      fp, 0(sp)       # salva fp precedente
        mv      fp, sp          # fp punta all'inizio del record di attivazione di SUM

        # calcola espressione
        mul     t0, a0, a1
        addi    a0, t0, 17

        addi    sp, fp, 8       # ripristina valore iniziale di fp usando fp
        lw      fp, 0(fp)       # ripristina fp dallo stack tramite fp
        jr      ra
```