UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Esempi di Programmi Assembly

Luigi Palopoli, Marco Roveri,
Luca Abeni

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Scopo della lezione

- In questa lezione vedremo alcuni esempi di programmi (o frammenti di programmi) in vari linguaggi assembly per renderci conto delle differenze
- Partiremo da assembly RISC-V e ARM
- Successivamente passeremo all'assembly INTEL

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Semplici istruzioni aritmetiche logiche

- Partiamo dal semplicissimo frammento che abbiamo visto a lezione

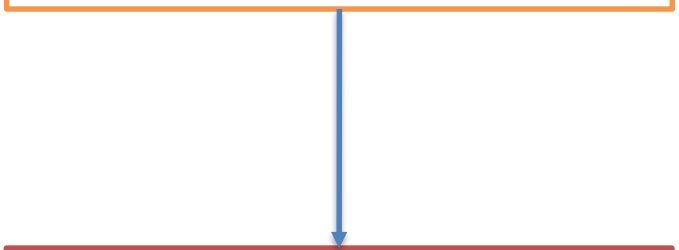
```
f = (g + h) - (i + j);
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Supponendo che g, h, i, j siano in x19, x20, x21, e x22, e che il risultato si voglia mettere in x23, la traduzione è semplicemente

```
f = (g+h)-(i+j);
```



```
add x5, x19, x20
add x6, x21, x22
sub x23, x5, x6
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V (v2)

- Supponendo che g, h, i, j siano in x19, x20, x21, e x22, e che il risultato si voglia mettere in x23, la traduzione è semplicemente

```
f = (g+h)-(i+j);
```

```
add x23, x19, x20
add x6, x21, x22
sub x23, x23, x6
```

In questa versione é usato un registro in meno:
Il risultato intermedio é memorizzato in x23

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Per INTEL, supponiamo che g, h, I, j siano in rdi, rsi, rcx, rdx e che il risultato si voglia in rax.
- Il problema è come fare la somma a due operandi e risultato in un terzo operando, cosa possibile usando l'istruzione lea

```
f = (g+h)-(i+j);
```



```
leaq (%rdi, %rsi), %rax     //rax = rdi + rsi
addq %rcx, %rdx             //rdx = rdx + rcx
subq %rdx, %rax             //rax = rax - rdx
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Accesso alla memoria

- Riguardiamo ancora l'esempio visto a lezione

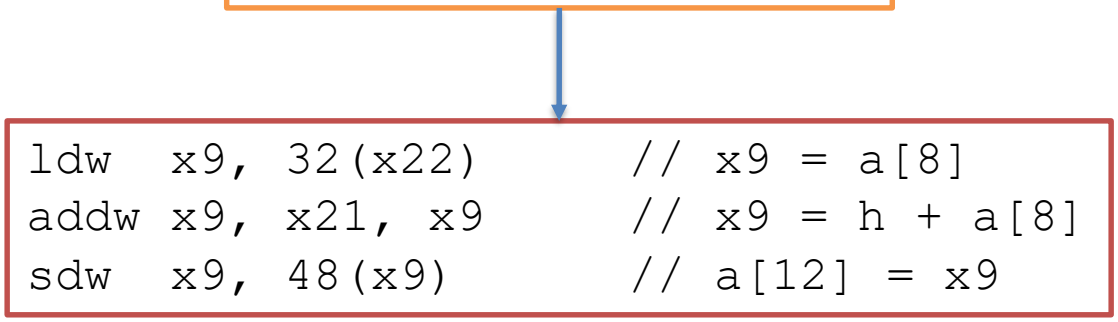
```
a[12] = h + a[8];
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Supponiamo che h sia in x21 e che il registro base del vettore a sia in x22

```
a[12]= h + a[8];
```



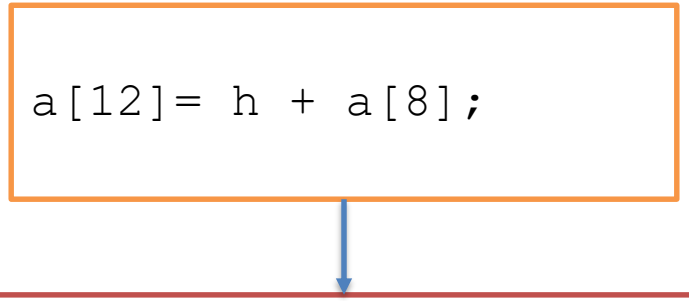
```
ldw   x9, 32(x22)      // x9 = a[8]
addw  x9, x21, x9      // x9 = h + a[8]
sdw   x9, 48(x9)       // a[12] = x9
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Supponiamo di avere h in edi e l'inidirizzo di a in rsi.
- Grazie al fatto di poter avere operandi in memoria stavolta ce la caviamo con due istruzioni

```
a[12]= h + a[8];
```



```
addl 32($rsi), %edi      //edi = edi + a[8]
movl %edi, 48(%rsi)      //a[12] = edi
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Blocchi condizionali

- Consideriamo il seguente blocco

```
if (i == j)
   f = g + h;
else
   f = g - h;
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Supponendo di avere f, g, h, i, j nei registri da x19 a x23 avremo

```
if (i == j)
    f = g + h;
else
    f = g - h;
```



```
    bne x22, x23, L2     // se x22 neq x23 vai a L2
    add x19, x20, x21    // x19 = g + h
    beq x0, x0, L3       // se x0 == x0 vai a L3
L2:
    sub x19, x20, x21    // v0 = g - h
L3:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Assumiamo che g, h, i, j siano in rdi, rsi, rcx, rdx e che f si voglia in rax

```
if (i == j)
    f = g+h;
else
    f = g - h;
```

```
    cmpq  $rcx, %rdx          // i == j??
    jne  L2
    leaq (%rdi, %rsi), %rax   //f = g + h
   jmp L3
L2:
   movq %rdi, %rax            //f = g
   subq  %rsi, %rax           //f = f-h. NOTARE COMPLICAZIONE
L3:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL (magia)

- Possiamo fare di meglio usando uno strano oggetto (move condizionale) che fino ad ora i compilatori avevano evitato di usare

```
if (i == j)
    f = g+h;
else
    f = g - h;
```

```
leaq    (%rdi, %rsi), %rax      //rax = g+h
subq    %rsi, %rdi              //rdi = g-h
cmpq    %rcx, %rdx
cmovne %rdi, %rax     //spostiamo se il cmp precedente NE
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Condizione con disuguaglianza

- Supponiamo ora di avere:

```
if (i < j)
   f = g + h;
else
   f = g - h;
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V (v2)

- Supponendo di avere f, g, h, i, j nei registri da X19 a x23 avremo

```
if (i < j)
    f = g + h;
else
    f = g - h;
```

```
    blt x22, x23, L2     // se x22 < x23 vai a L2
    sub x19, x20, x21    // f = g - h
    beq x0, x0, L3       // se x0 == x0 vai a L3
L2:
    add x19, x20, x21    // f = g + h
L3:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Assumiamo che g, h, i, j siano in rdi, rsi, rcx, rdx e che f si voglia in rax

```
if (i < j)
    f = g+h;
else
    f = g - h;
```

```
    cmpq  $rcx, %rdx          // i == j??
    jge  L2
    leaq (%rdi, %rsi), %rax   //f = g + h
   jmp L3
L2:
    movq %rdi, %rax           //f = g
   subq  %rsi, %rax           //f = f-h. NOTARE COMPLICAZIONE
L3:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL (magia)

- Ancora una volta possiamo usare la move condizionale

```
if (i < j)
    f = g+h;
else
    f = g - h;
```

```
leaq    (%rdi, %rsi), %rax      //rax = g+h
subq    %rsi, %rdi              //rdi = g-h
cmpq    %rcx, %rdx
cmovge %rdi, %rax      //spostiamo se il cmp precedente GE
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Ciclo while

- Consideriamo il seguente ciclo while

```
i = 0;
while (a[i]==k)
   i += 1;
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Supponendo di tenere i in x22, k in x24 e l'indirizzo base di a sia in x25

```
i = 0;
while (a[i] == k)
    i += 1;
```

```
    sd x22, x0              // i = 0
L1:
    slli    x10, x22, 2     // x10 = i * 4
    add     x10, x10, x25   // x10 = indirizzo di a[i]
    ldw      x9, 0(x10)      // x9 = a[i]
    bne     x9, x24, L2     // se a[i] != k vai a L2
    addi    x22, x22, 1     // i = i + 1
    beq     x0, x0, L1      // se 0 == 0 vai a L1
L2:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Stavolta è possibile sfruttare la potenza del CISC

```
i = 0;
while (a[i]==k)
    i += 1;
```

```
    cmpl   (%rsi), %edi      //%rsi piunta ad a
    jne    L2                //Ciclo mai eseguito
    movq   $0, %rax          //i = 0
L1:
    addq $1, %rax                 //i++
    cmpl %edi, (%rsi, %rax,4)  //k è a 32 bit
    je     L1
    jmp L3
L2: movq $0, %rax
L3:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Funzione Foglia

- Si definisce "foglia" una funzione che non ne chiama altre.
- Se non facciamo delle ottimizzazioni andremo a salvare (prologo) e ripristinare (epilogo) registry in maniera poco furba

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Esempio

```
int esempio foglia(int g, int h,
                   int i , int j) {
   int f;
   f = (g + h) - (i + j);
   return f ;
}
```

- Abbiamo una sola variabile locale (f) per la quale è possibile usare un registro

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Traduzione tenendo conto che g, h, i, j corrispondono ai registri da x10 a x13

```
int esempio_foglia(int g, int h,
                   int i , int j) {
   int f;
   f = (g + h)- (i + j);
   return f ;
}
```

```
esempio_foglia:
      addw    a0,a0,a1     //  f = g+h (a0 corrisponde a c10
      addw    a3,a2,a3    //   a3 = i+j
      subw    a0,a0,a3    //f = f – a3
      ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione non ottimizzata

Senza ottimizzazioni il risultato è piuttosto diverso

```
esempio_foglia:
    pushq   %rbp
    movq %rsp, %rbp
     movl    %edi, -20(%rbp)     //Prologo
     movl    %esi, -24(%rbp)
     movl    %edx, -28(%rbp)
     movl    %ecx, -32(%rbp)

     movl    -20(%rbp), %edx     //edx = h
     movl    -24(%rbp), %eax     //eax = g
     leal    (%rdx,%rax), %ecx   //f = ecx = g+h
     movl    -28(%rbp), %edx     //edx = i
     movl    -32(%rbp), %eax      //eax = j
     addl    %edx, %eax           //eax = i+j
     subl    %eax, %ecx          //ecx = g+h – (i+j)
     movl    %ecx, %eax          //aex = ecx

     movl    %eax, -4(%rbp)
     movl    -4(%rbp), %eax
     popq    %rbp
     ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Traduzione più ottimizzata

```
int esempio foglia(int g, int h, int i , int j) {
    int f;
    f = (g + h) - (i + j);
    return f ;
}
```

```
esempio_foglia:
leal    (%rdi,%rsi), %eax
    addl    %ecx, %edx
    subl    %edx, %eax
    ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Funzioni non foglia

- Consideriamo il seguente caso più complesso

```
int inc(int n)
{
   return n +1;
}
int f(int x)
{
   return inc(x) – 4;
}
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- La traduzione di inc è simile alla precedente traduzione, supponendo che n è in x10 (=a0) e risultato in x10

```
int inc(int n) {
    return n + 1;
}
```



```
inc:
    addiw   a0,a0,1
    ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- La traduzione di f richiede più attenzione. Supponiamo anche qui che n è in x10 (a0).

```
int f(int n) {
    return inc(n) - 4;
}
```

```
f:
    addi   sp, sp, -16          //Prologo
    sd     ra, 8(sp)

    jal    ra, inc
    addiw  a0, a0, -4

    ld     ra, 8(sp)            //Epilogo
    addi   sp, sp, 16
    ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- La traduzione INTEL è più semplice

```
int inc(int n) {
    return n + 1;
}
```



```
inc:
    leal    1(%rdi), %eax
    ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- La traduzione INTEL è più semplice poichè il salvataggio del return address è fatto in automatico con la call

```
int f(int n) {
    return inc(n) - 4;
}
```



```
inc:
    call   inc
    subl   $4, %eax
    ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Ordinamento di array

- Passiamo a qualcosa di più complesso. Un algoritmo noto come insert sort

```
void sposta(int v[], size_t i) {
  size_t j;
  int appoggio;

  appoggio = v[i];
  j = i - 1;
  while ((j >= 0) && (v[j] > appoggio)) {
    v[j+1] = v[j];
    j = j-1;
  }
  v[j+1] = appoggio;
}
```

```
void ordina(int v[], size_t n) {
   size_t i;
   i = 1;
   while (i < n) {
      sposta(v, i);
      i = i+1;
   }
}
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Cominciamo da sposta. Stavolta le cose sono più complesse. Assumiamo che i parametri siano memorizzati in x10, x11 rispettivamente. Usiamo x5, x6, x28 per j e appoggio.

```
void sposta(int v[], size_t i) {
  size_t j;
  int appoggio;

  appoggio = v[i];
  j = i - 1;
```

```
sposta:
        slli    a4,a1,2       //a4 = i*4
        add     a5,a0,a4            //a5 = &v[i]
        lw      a3,0(a5)            //a3 = v[i]
        addiw   a1,a1,-1      //a1 = a1-1 (i = j-1)
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Ciclo

```
while ((j >= 0) && (v[j] > appoggio)) {
   v[j+1] = v[j];
   j = j-1;
  }
```

```
        bltz    a1,.L2              //se j < 0 esci dal ciclo
        lw      a4,-4(a5)                 // a4 = v[i-1]=v[j]
        bge     a3,a4,.L2           //se appoggio è >= v[j] esci
        li      a2,-1               //carica -1 in a2
.L3:
        sw      a4,0(a5)                  //Memorizza v[j] (a4) in v[j+1)
        addiw   a1,a1,-1            //a1 = a1-1
        beq     a1,a2,.L4           // salta se a1 = -1
        addi    a5,a5,-4                  // j=j-1
        lw      a4,-4(a5)
        bgt     a4,a3,.L3                 //Salta se v[j] > appoggio
        j       .L2
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Uscita da sposta

```
 v[j+1] = appoggio;
}
```

```
.L4:
        li      a1,-1
.L2:
        addi    a1,a1,1
        slli    a1,a1,2
        add     a1,a0,a1
        sw      a3,0(a1)
        ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Passiamo ora alla funzione ordina. Assumiamo che i parametri siano memorizzati in x10, x11 rispettivamente

```
void ordina(int v[], size_t n) {
  size_t i;
  i = 1;
```

```
        li      a5,1
         ble     a1,a5,.L11
         addi    sp,sp,-32
         sd      ra,24(sp)
         sd      s0,16(sp)
         sd      s1,8(sp)
         sd      s2,0(sp)
         mv      s1,a1
         mv      s2,a0
         li      s0,1
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Passiamo al loop

```
while (i < n) {
   sposta(v, i);
   i = i+1;
}
```

```
.L8:
        mv      a1,s0
        mv      a0,s2
        call    sposta
        addiw   s0,s0,1
        bne     s1,s0,.L8
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Epilogo ordina

```
  ld      ra,24(sp)
        ld      s0,16(sp)
        ld      s1,8(sp)
        ld      s2,0(sp)
        addi    sp,sp,32
        jr      ra
.L11:
        ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

INTEL

- Riguardiamo lo stesso codice implementato tramite INTEL

```
void sposta(int v[], size_t i) {
  size_t j;
  int appoggio;

  appoggio = v[i];
  j = i - 1;
```

```
sposta:    # rdi = v, rsi = I
           # rax = j, r10d = appoggio
  movq %rsi, %rax
  movl (%rdi, %rax, 4), %r10d  # appoggio = v[i]
  dec %rax
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

INTEL

- Vediamo il ciclo

```
while ((j >= 0) && (v[j] > appoggio)) {
   v[j+1] = v[j];
   j = j-1;
 }
```

```
ciclo:
   cmpq $0, %rax                    // confronta 0 e rax
   jl   out                         // esci se j < 0
   movl (%rdi, %rax, 4), %r11d      // metti v[j] in %r11
   empl %r10d, %r11d                // confronta v[j] e appoggio
   jle  out                         // se v[j] < appoggio esci
   movl %r11d, 4(%rdi, %rax, 4)
   dec  %rax
   jmp ciclo
out:
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

INTEL

- Uscita da sposta

```
 v[j+1] = appoggio;
}
```



```
out:
    movl %r10d, 4(%rdi, %rax, 4)
    ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

INTEL

- Vediamo la procedura ordina, che non è foglia.
- Il salvataggio sullo stack è semplificato

```
void ordina(int v[], size_t n) {
  size_t i;
  i = 1;
```



```
ordina:                                 // rdi = v, rsi = n
                                        // rbx = i
    pushq %rbx
    movq $1, %rbx
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

INTEL

- Il loop

```
while (i < n) {
    sposta(v, i);
    i = i+1;
}
```

```
loop_ordina:
    cmp %rbx, %rsi
    jle out_ordina
    pushq %rsi
    movq %rbx, %rsi
    call sposta
    popq %rsi
    inc %rbx
    jmp loop_ordina
out_ordina
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

INTEL

- Epilogo

```
out_ordina
   popq %rbx
   ret
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Copia Stringhe

- Consideriamo ora

```
void copia_stringa(char d[], const char s[]) {
  size_t i = 0;

  while ((d[i] = s[i]) != `0`) {
    i += 1;
  }
}
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V

- Traduzione RISC-V

```
void copia_stringa(char d[], const char s[]) {
 size_t i = 0;
```

```
copia_stringa:
   addi sp, sp, -8      // aggiorna stack per inserire 1 element
   sd x19, 0(sp)        // salva x19
   add x19, x0, x0      // i = 0
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Loop

```
while ((d[i] = s[i]) != `0`) {
  i += 1;
}
```

```
LoopCopiaStringa:
    add x5, x19, x11                 // indirizzo di s[i]
    lbu x6, 0(x5)                    // x6 = s[i]
    add x7, x19, x10                 // indirizzo di d[i]
    sb x6, 0(x7)                     // d[i] = s[i]
    beq x6, x0, LoopCopiaStringaEnd  // se`0` vai a LoopCopiaStringaEnd
    addi x19, x19, 1                 // i += 1
    jal x0, LoopCopiaStringa         // salta a LoopCopiaStringa
LoopCopiaStringaEnd
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione RISC-V continua

- Chiusura

```
LoopCopiaStringaEnd
    ld x19, 0(sp)     // ripristina contenuto di x19
    addi sp, sp, 8    // aggiorna lo stack eliminando un element
    jral, x0, 0(x1)   // ritorna al chiamante
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Inizio

```
void copia_stringa(char d[], const char s[]) {
  size_t i = 0;
```



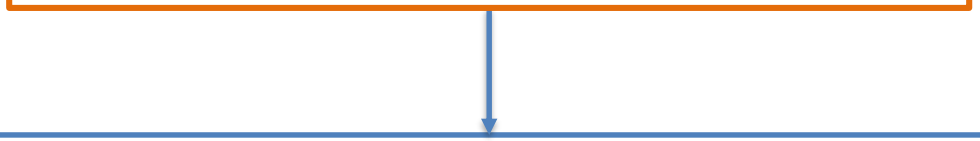
```
copia_stringa:
    movzbl (%rsi), %eax
    movb %al, (%rdi)
    testb  %al, %al
    je L1
    movl $0, %eax
```

UNIVERSITÀ DEGLI STUDI DI TRENTO
Dipartimento di Ingegneria
e Scienza dell'Informazione

Traduzione INTEL

- Loop

```
while ((d[i] = s[i]) != `0`) {
  i += 1;
}
```



```
L3:
    addl $1, %eax
    movslq %eax, %rcx
    movzbl (%rsi, %rcx), %edx
    movb %dl, (%rdi, %rcx)
    testb %dl, %dl
    jne L3
L1:
    ret
```