Confronto compilatori

1 Introduzione

Confronto i compilatori. I Compilatori confrontati sono $gcc\ RISCV,\ gcc\ x86_64$ e $gcc\ ARM$. I compilati vengono compilati con il livello di ottimizzazione di default (-O0).

2 Operazioni

2.1 Addizione con costante

```
1 | int get_num(int num) {
2      return 23 + num;
3 | }
```

La funzione è una semplice funzione scritta in C che dato un numero di tipo intero restituisce il numero sommato a 23.

RISC-V

Il sorgente compilato con il compilatore RISC-V(a) da riga 2 fino a riga 6 predispone la chiamata della procedura posizionando sullo stack il necessario, da riga 7 inizia la funzione. Su quella riga viene recuperato il valore di num che alla riga 8, tramite l'operazione di add immediate, viene sommato a num. Il risultato dell'operazione addiw è la somma del valore di num sommato alla costante 23, il risultato è esteso su 64 bit, vengono ignorati gli errori di overflow. Successivamente tramite la pseudo istruzione sext.w che prende i 32 bit inferiori e li memorizza nel registro rd. Questa istruzione corrisponde a addiw rd, rs,1 0. Il risultato viene spostato nel registro a0 che, nei processori RISC-V, viene utilizzato come restituzione di risultato. Le righe successive ripristinano lo stack e restituice il controllo al chiamante.

```
addi
                                                                       get_num:
        s0.24(sp)
                                                                                push
                                                                                         {r7}
sd
                                                                                         sp, sp, #12
        s0,sp,32
mν
        a5,a0
                                        push
                                                                                add
                                                                                             sp, #0
[r7, #4]
        a5,-20(s0)
sw
                                        mov
                                                                                str
         a5,-20(s0)
                                                 DWORD PTR [rbp
                                                                                         r3, r3,
r0, r3
addiw
        a5,a5,23
                                                                                adds
                                                 eax, DWORD PTR
        a5, a5
sext.w
                                        mov
                                                                                mov
                                                                                adds
1d
        s0,24(sp)
                                        add
                                                 eax. 23
                                                                                mov
                                                                                         r7, [sp], #4
addi
        sp, sp, 32
                                        pop
                                                 rbp
                                                                                ldr
(a) RISC-V
                                           (b) x86
                                                                                  (c) ARM
```

Figure 1: Funzione di somma

x86

Il sorgente (b) è compilato con gcc di x86. Da riga 2 fino a riga 4 viene preparato lo stack, a riga 5 viene posizionato num nel registro eax che a riga 6 viene sommato a 23 che viene memorizzato nel registro eax. Infine viene ridato il controllo al chiamante.

ARM

Il sorgente (c) , compilato con gcc ARM, mostra che la preparazione della procedure si esegue da riga 2 a riga 5, le con le due righe successive si esegue la funzione. La riga 6 recupera il valore di num la riga successiva calcola il valore del risultato e, infine, alla riga 8 si sposta il risultato nel registro di restituzione

2.2 Addizione

```
1 int sumGen(int num, int num2, int num3) {
2    return num + num2 + num3;
3 }
```

Inserisci sorgenti Addizione generale

2.3 Moltiplicazione

```
1 int mult2(int num){
2     return 2 * num;
3 }
```

La funzione dato un numero di tipo intero restituisce il numero moltiplicato per 2. Per i sorgenti le parti di preparazione sono simili per le rispettive preparazioni precedenti.

Moltiplicazioni per potenze di 2

```
addi
                    s0.24(sp)
            sd
                                                                                 mult2:
                    s0,sp,32
                                          mult2:
            mν
                    a5,a0
                                                   push
                                                                                          push
                    a5,-20(s0)
            sw
                                                   mov
                                                                                          mov
                                                                                                  rbp, rsp
            lw
                     a5,-20(s0)
                                                            DWORD PTR [rbp
                                                                                                  DWORD PTR [rbp
                                                -4], edi
            slliw
                    a5,a5,1
                                                                                       -4], edi
                                                            eax, DWORD PTR
                                                                                                   eax, DWORD PTR
                    a5, a5
            sext.w
                                                   mov
                                                                                          mov
10
                     a0,a5
                                                                                        [rbp-4]
            1d
                    s0,24(sp)
                                                   add
                                                            eax. eax
                                                                                          add
                                                                                                   eax. eax
12
                    sp, sp, 32
            addi
                                                   pop
                                                            rbp
                                                                                          pop
                                                                                                  rbp
            (a) RISC-V
                                                      (b) x86
                                                                                            (c) ARM
```

Figure 2: Moltiplicazione per 2

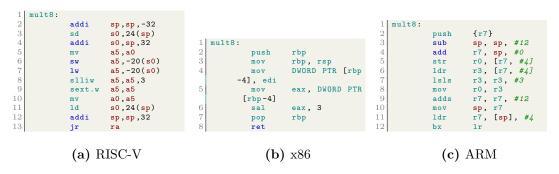


Figure 3: Moltiplicazione per 8

Nel sorgente RISC-V (2.a) l'operazione di moltiplicazione per 2 avviene tramite uno shift logical left di 1 bit (SLLIW). Stesso concetto avviene nel sorgente ARM (2.c) dove l'operazione di moltiplicazione per 2 avviene tramite lo shift left di 1 bit. Invece nel sorgente x86(2.b) la moltiplicazione avviene tramite una somma. Questa somma è un caso particolare, infatti se volessimo moltiplicare per una qualsiasi potenza di 2 (ad esempio in figura 3) le operazioni avvengono tutte tramite shift left di un opportuno valore.

Moltiplicazioni per non potenze di 2

Nel caso più generale i sorgenti si comportano in maniera molto diversa. Partendo dal sorgente ARM (4.c) il corpo della funzione è tra le righe 6, 7 e 8 dove, dopo aver recuperato il valore di num, attraverso l'operazione di mul viene calcolato il valore del prodotto tra num e 11(la costante). Nel sorgente x86 (4.b) invece viene calcolato attraverso shift e sommando dei registri con i risultati (riga 5-10). Infine nel sorgente RISC-V (4.a) la moltiplicazione

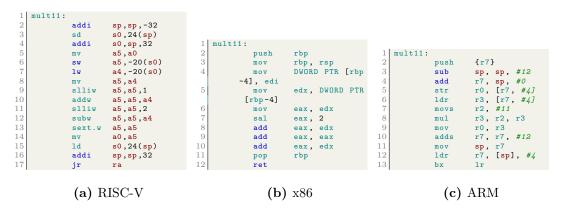


Figure 4: Moltiplicazione per 2

avviene attraverso shift e addizioni e sottrazioni (riga 7-13).

?Possibile? inserimento di sottrazione e divisione Inserisci operazione di shift