

Calcolatori Elettronici

Esercitazione 3

M. Sonza Reorda – M. Monetti

M. Rebaudengo – R. Ferrero

L. Sterpone – E. Vacca

Politecnico di Torino

Dipartimento di Automatica e Informatica

Obiettivi

- Input robusto
- Operazioni di moltiplicazione e divisione

Moltiplicazione

mul Rdest, Rsrc, Src Signed multiply with **no overflow**
Sets: $Rdest = Rsrc * Src$ or Imm

mulu Rdest, Rsrc, Src Unsigned multiply with **no overflow**
Sets: $Rdest = Rsrc * Src$ or Imm

mulo Rdest, Rsrc, Src Signed multiply with overflow
Sets: $Rdest = Rsrc * Src$ or Imm

mulou Rdest, Rsrc, Src Unsigned multiply with overflow
Sets: **\$lo** = $Rsrc * Src$ or Imm

mult Rsrc1, Rsrc2 Signed 64-bit multiply
Sets **\$hi:\$lo** = $Rsrc * Src$ or Imm

multu Rsrc1, Rsrc2 Unsigned 64-bit multiply
Sets **\$hi:\$lo** = $Rsrc * Src$ or Imm

MOVE con registri &hi e \$lo

mfhi Rd Rd = **\$hi**

mflo Rd Rd = **\$lo**

- NON supportato da simulatore QtSpim

Esercizio 1

- La system call 5 permette di leggere in input un numero intero con segno.
- Cosa succede se l'utente introduce da tastiera un carattere non numerico?
- Si realizzi un programma per effettuare una lettura robusta di un numero intero unsigned.
- Il programma legge singoli caratteri tramite la system call 12, verifica se sono cifre e termina quando l'utente preme '\n' (invio).

Esercizio 2

- Si modifichi l'esercizio precedente per la lettura robusta di un numero intero unsigned tramite la system call 12.
- Oltre a verificare se i caratteri introdotto siano cifre, il programma deve controllare se il numero sia rappresentabile su 4 byte.
- Il programma termina quando è letto '\n'; il numero introdotto in input deve essere stampato a video tramite la system call 1.

Implementazione

- Per convertire una sequenza di caratteri in un intero si utilizza un ciclo. Dopo aver inizialmente azzerato un registro accumulatore, ad ogni iterazione:
 1. l'ultimo carattere letto è convertito in intero sottraendo al suo codice ASCII il valore '0'
 2. il valore nell'accumulatore è moltiplicato per 10
 3. si somma il valore calcolato all'accumulatore.
- Si noti che le operazioni ai punti 2 e 3 possono dare un *overflow*. In questo caso il programma deve stampare un opportuno messaggio.

Esempio

- L'utente inserisce i caratteri '3', '4', '6', '\n'
- Prima iterazione:
 1. valore letto = '3' - '0' = 3
 2. accumulatore * 10 = 0 * 10 = 0
 3. valore corrente = 0 + 3 = 3
- Seconda iterazione:
 1. valore letto = '4' - '0' = 4
 2. accumulatore * 10 = 3 * 10 = 30
 3. valore corrente = 30 + 4 = 34
- Terza iterazione:
 1. valore letto = '6' - '0' = 6
 2. valore precedente * 10 = 34 * 10 = 340
 3. valore corrente = 340 + 6 = 346
- Quarta iterazione:
 1. valore letto = '\n'
 2. Il programma termina e stampa a video 346

Verifica dell'overflow in binario puro

- `MUL` effettua la moltiplicazione tra i due operandi, restituisce il risultato su 32 bit e non scatena eccezione in caso di overflow
- Per verificare l'overflow su 32 bit in binario puro si può usare:
 - `MULOU`: scatena un'eccezione
 - `MULTU`: calcola il risultato su 64 bit, nei registri `$hi` e `$lo`. Se `$hi` è diverso da zero, c'è overflow

Verifica dell'overflow in Ca2

- MUL non scatena eccezione in caso di overflow (non c'è distinzione tra signed e unsigned forzando il risultato su 32 bit)
- Per verificare l'overflow su 32 bit in Ca2 si usa:
 - MULO: scatena un'eccezione
 - MULT: calcola il risultato su 64 bit, nei registri \$hi e \$lo. Non c'è overflow su 32 bit se:
 - \$hi è uguale a 0 e il bit più significativo di \$lo è 0
 - \$hi è 0xFFFFFFFF e il bit più significativo di \$lo è 1

Esercizio 3

- Siano date tre variabili di tipo *byte* in memoria, che rappresentino rispettivamente il numero di giorni, ore e minuti passati da un certo istante T_0 .
- Si calcoli il numero totale di minuti passati da T_0 , e tale valore sia salvato nella variabile di tipo *word* risultato.
- È possibile ottenere *overflow* durante i calcoli?

Esercizio 4

- Si scriva un programma che acquisisca DIM valori *word* e quindi ne calcoli la media (intera) e la stampi a video.
 - DIM deve essere dichiarato come costante
 - Si lavori nell'ipotesi di non avere *overflow* nei calcoli
 - Si noti il tipo di arrotondamento effettuato sul risultato della divisione.