Run-Time Error 200 Divide by 0?

La soluzione (definitiva) a un bug ormai storico della unit CRT dell'altrettanto storico compilatore Borland per il linguaggio Pascal

di Antonino Calderone

Il linguaggio Pascal ha rappresentato un ottimo investimento per le scuole e sovente per le università, essendo accreditato come uno degli strumenti più utili in ambito didattico/informatico. Tra gli ambienti di sviluppo più conosciuti e apprezzati dalla comunità dei programmatori DOS, un posto di rilievo è stato certamente occupato dai prodotti Borland. La diffusione avuta da questi strumenti è dovuta al favore incontrato da parte degli utenti, che ne hanno apprezzato soprattutto facilità d'uso e disponibilità di ampia documentazione (anche cartacea).

I compilatori di Borland hanno rappresentato (e continuano a rappresentare) un riferimento importante. La credibilità e la qualità dei prodotti di questa software house non sono in discussione. Proprio per questo molta perplessità ha destato un inaspettato problema con programmi scritti in Turbo o Borland Pascal (non saprei dire da quale versione, ma certamente la 7 ne era affetta), ma ancora di più il fatto che da parte di Borland non fosse stata prodotta nessuna patch per porre rimedio allo stesso. Molti dei programmi realizzati con questo compilatore manifestavano (e manifestano) un problema a run-time, non appena in esecuzione, su certi calcolatori. L'applicativo conclude immediatamente la propria esecuzione dopo l'emissione di un messaggio del tipo "*Runtime error 200*".

La unit CRT

Il bug col quale abbiamo a che fare si manifesta solo su macchine veloci, di nuova generazione (ma neanche poi tanto, perché è ravvisabile già su Pentium Pro con frequenze di clock di 200 Mhz).

Antonino Calderone acalderone@infomedia.it

Lavora come software engineer presso Marconi Communications SpA nella divisione Optical Networks. Inoltre collabora con la DigiFacta SW Engineering S.r.l. Si occupa principalmente di sistemi e protocolli per le telecomunicazioni, firmware, device driver, internals di sistemi operativi.

Il problema non affligge i programmi compilati per l'ambiente Windows, perché il frammento di codice incriminato si trova in una unit (denominata CRT) che fa parte della DOS Run-Time Library (RTL) contenuta all'interno dei file di libreria *TURBO.TPL* oppure nel file *TPP.TPL*, rispettivamente usati per la compilazione di codice destinato a funzionare in modalità DOS reale e in modalità DOS protetta.

La unit CRT contiene le funzioni necessarie per il controllo dello schermo in modalità testo, il supporto per i codici estesi per l'input da tastiera, il supporto per la generazione dei suoni e dei ritardi.

La funzione Delay

Una delle funzioni dichiarate nella unit CRT è la funzione *Delay* il cui prototipo è

```
procedure Delay(MS: Word);
```

Questa funzione è usata per generare ritardi compresi tra 1 e 65535 millisecondi.

In ambiente DOS un modo standard per generare un ritardo è quello di servirsi del conteggio dei timer tick. La locazione di memoria che tiene traccia dei timer tick è una *DWORD* che si trova all'indirizzo

LISTATO 1 Loop usato per stimare il numero di iterazioni fatte in un intervallo di 55 ms e per generare ritardi con granularità di un millisecondo

```
;Listato 1

DelayLoop:
@@1: SUB AX,1
     SBB DX,0        ; Decrementa DX:AX
     JC @@2          ; DX:AX < 0 ? Si fine del ciclo
     CMP BL,ES:[DI] ; Timer tick invariato ?
     JE @@1          ; Si: ripeti da @@1, no fine del ciclo
@@2: RET
```

LISTATO 2 Invocazione della routine DelayLoop in fase di start up della unit CRT

```
;…
     MOV ES,Seg0040      ; ES:DI -> Timer tick
     MOV DI,OFFSET Timer
     MOV BL,ES:[DI]      ; Leggi il contatore dei timer
                                                tick
@@2: CMP BL,ES:[DI]      ; È stato incrementato ?
     JE @@2              ; No, attendi.
     MOV BL,ES:[DI]      ; Si, Salva in BL l'ultimo
                                              valore
     MOV AX,-28          ; del timer letto
     CWD                 ; Assegna a DX:AX OFFFFFFE4h
     CALL DelayLoop

;… (continua nel listato 3)
```

assoluto 0x0046C. Questo valore è incrementato dal sistema 18.2 volte al secondo (vale a dire ogni 55 ms) in risposta a un interrupt hardware (*IRQ 0*) generato dal timer di sistema.

Il compito di contare i timer tick è assolto dall'ISR dell'*INT 8* (così viene rimappato l'*IRQ 0*).

Conteggio dei timer tick

Un esperimento che può essere condotto, sulla base di quanto è stato appena detto, è quello che vi proponiamo di seguito.

Si mandi in esecuzione il *DEBUG* del DOS (anche se si sta lavorando in ambiente Windows).

Si lanci più volte il comando *d 40:6C 6F* e si osservi che ogni volta che viene ripetuto, il contenuto della locazione di memoria 0x0046C cambia, in particolare aumenta (come era lecito aspettarsi).

Per esaminare il codice dell'ISR dell'*INT 8*, responsabile del conteggio dei timer tick, si proceda nel seguente modo. Si lanci il comando *a 100*.

Si inseriscano le seguenti righe di codice assembly

```
xor ax, ax
mov ds, ax
mov bx, 20
jmp far [bx]
```

Si lanci, il comando *g 107* e di seguito il comando *t*.

A questo punto i registri *CS:IP* dovrebbero puntare la prima istruzione della routine che gestisce il timer interrupt. Per visualizzare il codice Assembly di questa routine si usi il comando *u*.

Scorrendo il listato così ottenuto, si dovrebbe individuare un frammento di codice simile a quello illustrato di seguito

```
…
0212:074D B84000      MOV AX,0040
0212:0750 8ED8        MOV DS,AX
0212:0752 33C0        XOR AX,AX
0212:0754 8EC0        MOV ES,AX
0212:0756 FF066C00    INC WORD PTR [006C]
0212:075A 7504        JNZ 0760
0212:075C FF066E00    INC WORD PTR [006E]
…
```

LISTATO 3 La divisione del valore dei registri DX:AX per 55 usata per determinare il numero di iterazioni eseguibili in un millisecondo

```
;… (continua dal listato 2)
     NOT AX
     NOT DX
     MOV CX,55
     DIV CX
     MOV DelayCnt,AX ; DelayCnt = not(DX:AX) DIV 55
;…
```

che guarda caso è quello che incrementa il contatore dei timer tick.

Generazione di ritardi inferiori a 55 ms

Se la funzione Delay si basasse (solo) sul conteggio dei timer tick non potrebbe generare correttamente intervalli di durata inferiore a 55 ms.

Per ovviare a questo problema, nel codice di inizializzazione della Unit CRT è stata inserita una routine (*DelayLoop*) che assolve una duplice funzione:

1) stimare il numero di iterazioni fatte da un loop in un intervallo di 55 ms;
2) generare un ritardo di un millisecondo.

LISTATO 4 Il codice assembly equivalente alle modifiche derivanti dall'applicazione della patch fatte attraverso l'utility TPPATCH

```
;…
     CALL DelayLoop
     NOT AX
     NOT DX
     MOV CX, 37h          ; = 55 in decimale
     CMP DX,CX            ; DX < 37h ?
     JB @1                ; Si, fai la divisione
     MOV AX,0FFFFh        ; No, assegna a DelayCnt 65535
     JMP @2               ; Non eseguire la divisione
@1: DIV CX
@2: MOV DelayCnt, AX
;…
```

Run Time Error 200 'Divide by 0'.

Applications that use the CRT unit may generate this error message when running on very fast machines (i.e. Pentium Pro 180 and above). The cause of this error is a timing loop that occurs as part of the initialization of the CRT unit. This timing loop counts how many clock ticks occur within the loop and then that number is divided by 55. The result of this division is a value that is too large to fit into an integer value.

The 'Divide by 0' error message is the catch-all error that is displayed when this overflow occurs.

There are currently no Inprise endorsed patches for this problem. There are several user provided patches available on the internet that patch both the CRT unit as well as existing EXE files. The easiest way to obtain these patches is to go to www.altavista.digital.com and search on

'+ bp7patch.zip + tppatch.zip' without the quotes.

These patches are not endorsed or supported by Inprise and are used at your own risk.

RIQUADRO 1 Le informazioni sul problema, riportate al tempo sul sito Inprise (Borland)

Per illustrare come ciò venga fatto si esamini la definizione di *DelayLoop* (**Listato 1**) nel codice sorgente della unit CRT (questo codice, originariamente privo di commenti, è tratto dal file CRT.ASM fornito con i sorgenti della RTL versione 6.0 di Borland).

Quando *DelayLoop* è invocata dal codice di start up della unit CRT, la coppia *ES:DI* punta l'indirizzo [0040:006C], il registro *BL* è inizializzato col byte di ordine basso della *DWORD* aggiornata dal timer interrupt. La coppia di registri *DX:AX* è inizializzata con il valore 0xFFFFFFE4 (**Listato 2**). La *DWORD DX:AX* viene usata per calcolare il valore della variabile *DelayCnt*. Il complemento di *DX:AX* viene diviso per 55 e il quoziente viene salvato in *DelayCnt* come illustrato nel **Listato 3**. Quando *DelayLoop* è invocata dalla funzione *Delay* i valori dei registri usati come parametri sono impostati in maniera diversa: *DX* viene azzerato; *AX* è inizializzato con il valore di *DelayCnt*. Il valore di *BL*, e quello puntato da *ES:DI* rimangono costanti (*ES:DI* = 0040:0000), quindi il ciclo della routine *DelayLoop* è influenzato solo dal test su *AX*, e di conseguenza termina esattamente dopo *DelayCnt* iterazioni che avvengono proprio in un millisecondo.

Run-Time Error 200

Il meccanismo descritto nel precedente paragrafo, apparentemente funzionante, non tiene conto di un piccolo ma importante dettaglio. L'istruzione impiegata per calcolare *DelayCnt*, cioè l'istruzione

LISTATO 5 Modifiche alla dichiarazione della variabile DelayCnt

```
; Local workspace
;DelayCnt DW ?     ; Codice originale
DelayCnt DD ?     ; Adesso DelayCnt è dichiarato come DWORD
```

DIV CX (**Listato 3**) in particolari condizioni può sollevare una eccezione *INT 0*.

L'istruzione *DIV* genera un'eccezione *INT 0* in uno dei due casi: se il divisore (l'operando esplicito) è zero, oppure se il risultato della divisione è troppo grande per essere contenuto nel registro designato (in questo caso *AX*).

L'errore di *Run-Time 200* segnalato dai programmi che manifestano il bug si verifica quando la *DWORD* contenuta in *DX:AX* (usata per calcolare *DelayCnt*) diviene "troppo" grande. Per la precisione, più grande di 3.604.425 (0x36FFC9 in esadecimale) calcolato come risultato del prodotto di 65535 (valore massimo assegnabile al registro *AX*) per 55.

Con processori di vecchia generazione, la routine *DelayLoop* in 55 ms non supera la soglia di 3.604.425 iterazioni, mentre con i nuovi e veloci processori questo limite è abbondantemente oltrepassato, da cui il bug.

Mettiamoci una "patch"

Come ogni problema che si rispetti, anche per questo è possibile adottare più soluzioni.

Parecchi sviluppatori sono immediatamente corsi ai ripari producendo delle patch per porre rimedio al problema.

Alcune di queste patch intervengono direttamente sui file già compilati. Nel caso in cui non si disponga dei sorgenti o del compilatore non rimangono molte altre possibilità di intervento. Altre direttamente sui file di libreria con estensione *TPL*.

All'URL[2] si possono prelevare alcuni di questi programmi, come per esempio *TPPATCH*, scritto da Andreas Bauer (andi.tio@hit.handshake.de). *TPPATCH* individua direttamente sul file eseguibile il frammento di codice bacato e lo modifica nel modo illustrato nel **Listato 4**:

La correzione funziona in buona parte dei casi anche se presenta alcuni inconvenienti:

- Nel caso in cui *DX* = 36 e *AX*> 0xFFC9 il test non evita l'eccezione *INT 0*.
- Su macchine veloci i ritardi non sono calcolati correttamente; gli eventuali ritardi generati con *Delay* durano meno del dovuto.
- I byte in eccesso, introdotti dalla patch, sono recuperati grazie all'ottimizzazione di alcune istruzioni

di inizializzazione che precedono il frammento di codice modificato. Tuttavia la variabile globale *CheckEOF* (di tipo *Boolean*) non viene più inizializzata a 0 (*false*): l'istruzione MOV che opera questa inizializzazione viene sovrascritta dalla patch stessa.

- Corregge solo programmi compilati con la versione 7 (e forse 6) di TP o BP.

Modifichiamo la unit CRT usando codice Assembly 386

Una soluzione più efficace, ma ovviamente non sempre praticabile, consiste nella modifica e nella ricompilazione della unit CRT, e quindi nella ricompilazione dei programmi che usano questa unit.

Ai programmatori che in passato abbiano usato il TP (o il BP) per realizzare programmi DOS che ancora oggi vogliono supportare, è caldamente consigliato questo approccio. Illustriamo adesso come sia possibile modificare in Assembly 386 alcune parti della unit CRT (agendo direttamente sul codice sorgente contenuto nel file *CRT.ASM*) per eliminare il bug e consentire un corretta generazione dei ritardi con la funzione *Delay*.

Chiariamo subito che la scelta di impiegare codice 386 non è dettata dalla necessità (si poteva tranquillamente impiegare codice 8086/88!), ma dall'opportunità: siamo interessati a far funzionare correttamente il codice ricompilato su PC di nuova generazione.

Ci sembra sensato non interessarci alla compatibilità verso vecchie architetture, a fronte degli indubbi vantaggi derivanti dall'impiego di codice a 32 bit.

La prima modifica riguarda la nuova dichiarazione della variabile *DelayCnt* (**Listato 5**).

LISTATO 6 Riscrittura della routine DelayLoop e di parte della routine Delay

```
Delay:
     MOV BX,SP        ; Preleva dallo stack il valore
     MOV CX,SS:[BX+4] ; passato a Delay come argomento
     JCXZ @@2         ; Se questo valore è zero, termina
     MOV ES,Seg0040
     XOR DI,DI
     MOV BL,ES:[DI]   ; BL = [0040:0000]

;@@1: MOV AX,DelayCnt ;
;     XOR DX,DX       ; Codice originale
;     CALL DelayLoop  ;

@@1: CALL DelayLoop   ; Chiama DelayLoop
     LOOP @@1         ; per CX volte

@@2: RETF 2

; Delay one timer tick or by CX iterations

DelayLoop:
;@@1: SUB AX,1       ;
;     SBB DX,0       ; Codice
;     JC @@2         ;
;     CMP BL,ES:[DI] ; originale
;     JE @@1         ;
;@@2: RET            ;

      XOR EAX, EAX      ; Azzera EAX (usato come con
                                              tatore)
@@1:  CMP EAX, DelayCnt ; EAX == DelayCnt ?
      JE @@2            ; Si, termina il ciclo
      INC EAX           ; No, incrementa EAX
      CMP BL, ES:[DI]   ; È trascorso un timer tick ?
      JE @@1            ; No, Ripeti da @@1.
@@2:  RET               ; Si, fine del ciclo
```

LISTATO 7 Modifiche alla parte di codice che opera l'inizializzazione e infine la divisione per 55

```
Delay:
     MOV BX,SP        ; Preleva dallo stack il valore
     MOV CX,SS:[BX+4] ; passato a Delay come argomento
     JCXZ @@2         ; Se questo valore è zero, ter-
                                                 mina
     MOV ES,Seg0040
     XOR DI,DI
     MOV BL,ES:[DI]   ; BL = [0040:0000]

;@@1: MOV AX,DelayCnt ;
;     XOR DX,DX       ; Codice originale
;     CALL DelayLoop  ;

@@1: CALL DelayLoop   ; Chiama DelayLoop
     LOOP @@1         ; per CX volte

@@2: RETF 2

; Delay one timer tick or by CX iterations

DelayLoop:
;@@1: SUB AX,1       ;
;     SBB DX,0       ; Codice
;     JC @@2         ;
;     CMP BL,ES:[DI] ; originale
;     JE @@1         ;
;@@2: RET            ;

      XOR EAX, EAX      ; Azzera EAX (usato come con-
                                              tatore)
@@1:  CMP EAX, DelayCnt ; EAX == DelayCnt ?
      JE @@2            ; Si, termina il ciclo
      INC EAX           ; No, incrementa EAX
      CMP BL, ES:[DI]   ; È trascorso un timer tick ?
      JE @@1            ; No, Ripeti da @@1.
@@2:  RET               ; Si, fine del ciclo
```

Quindi la riscrittura della routine *DelayLoop* e di parte di *Delay* (**Listato 6**). E infine la modifica del codice di inizializzazione della unit CRT (**Listato 7**). Le modifiche apportate alla unit CRT non introducono novità strutturali.

Il nuovo codice opera fondamentalmente come il codice originale. Gli accorgimenti dipendono essenzialmente dall'impiego dei registri a 32 bit e dunque di istruzioni 386.

La funzione *DelayLoop* viene impiegata esattamente come prima, se si eccettua che non decrementa una coppia di registri a 16 bit ma ne incrementa uno solo a 32. Sono state modificate opportunamente la funzione *Delay* e una parte del codice di inizializzazione.

L'ultima parte del **Listato 7** contiene la modifica più importante, perché elimina la possibilità che in questo contesto si ripresenti una eccezione di tipo *INT 0*.

Il quoziente dell'istruzione *DIV ECX*, infatti, può essere sempre contenuto in *EAX*. In [1] si possono trovare ulteriori informazioni sull'istruzione *DIV*.

La ricompilazione della RTL

Una volta apportate le modifiche al file *CRT.ASM*, si può usare l'utility *MAKE* per ricompilare la unit CRT e ricreare i file *TURBO.TPL* e *TPP.TPL*. Questi file dovranno rimpiazzare i file originali, che si trovano nella directory *\BP\BIN*, assumendo *\BP* il percorso di installazione del compilatore.

Conclusioni

Vogliamo concludere con le informazioni prelevate direttamente dal sito di Borland (al tempo della nostra "recensione" si chiamava Inprise!) nella pagina di supporto per gli sviluppatori Pascal. Ve le riproponiamo in versione integrale nel **Riquadro 1**. Buon divertimento.

Bibliografia & Riferimenti

[1] Intel Corporation – "*80386 Programmer's Reference Manual*", Intel Corporation, 1988
[2] http://www.merlyn.demon.co.uk/pas-r200.htm