7

Die Verwendung des modernen Operationscodes

Für viele Programmierer stellt das strukturierte Programmieren eine Lebensart dar. Neue strukturbasierte und objektorientierte Sprachen wie zum Beispiel ADA, C++ und Modula-2 werden immer beliebter. PL/I, die eine der strukturiertesten Programmiersprachen ist, wurde von IBM während der frühen 60er Jahre entwickelt und in die Datenverarbeitungsgemeinde im Jahr 1966 eingeführt. PL/I unterstützt mehr strukturierte Programmkonstrukte als jede andere Sprache. PL/I hatte jedoch nie einen starken Stand in der Datenverarbeitungsgemeinde erreicht. Die ge-

nauen Gründe dafür sind sehr umstritten, aber die gängigste Spekulation ist, dass sie einfach zu groß ist. PL/I hat versucht, zu viel zu tun – um alles für alle zu sein. Zu dieser Zeit waren magnetische Medien neu und nicht so dicht beschreibbar, so dass eine so ”große” Sprache einfach nicht praktisch war.

RPG wurde auch in den frühen 60er Jahren entwickelt. Auch sie enthält die meisten – aber sicherlich nicht alle – strukturierten Programmierkonstrukte. Es gibt jedoch große Unterschiede zwischen PL/I und RPG. Ein grundlegender Unterschied besteht in der Größe: RPG ist eine kleine Sprache. Ihr Code kann als ”eng” angesehen werden. Sie wurde für kleine diskettenlose Maschinen wie die 1400 und 360 Modell 20 entwickelt.

Dieses Kapitel veranschaulicht die Verwendung des RPG-Operationscodes in der modernen RPG-Sprache. Die strukturierten Konstrukte, wie zum Beispiel IF, THEN, ELSE, WHEN, DO, DO WHILE und DO UNTIL decken den Funktionsumfang zusätzlich zu den anderen Operationen ab. Strukturierte Operationen unterstützen die Top-Down-Methode für Design und strukturierte Programmentwicklung. Zusätzlich wird auch ein Vergleich der modernen RPG-Sprache mit dem traditionellen RPG-Codieren geliefert - für erfahrene RPG-Programmierer, die die moderne RPG-Sprache erlernen.

Jedes Beispiel in diesem Kapitel kann als Aufgabe innerhalb eines größeren Programms oder Systems angesehen werden. Manche mögen voll funktional sein, während andere eine spezifischere Operation veranschaulichen.

Die Optimierung

Das Optimieren von Anwendungscode ist ein beliebtes Thema. Ironischerweise hat sich die Optimierung mehr auf die Leistung des Kompilers als auf die Anwenderprogramme konzentriert. Anwendungsprogrammierer sollten sich jedoch mehr um die Leistungsfähigkeit der Anwendung kümmern als um die Vollkommenheit des Kompilers.

In RPG kann die Verwendung von Bezugszahlen zu einer schlechten Performance führen. Ein Bedingungsanzeigereintrag in der Rechenbestimmung erzeugt einen "Vergleichs"-Befehl für jede verwendete Bezugszahl. Deshalb erzeugt jede Bezugszahl, die verwendet wird, um eine Operation zu konditionieren, einen unabhängigen Vergleichsbefehl.

Daraus könnte man schließen, dass sich die Verwendung von RPG-Indikatoren negativ auf die Leistung auswirkt. In der Tat ist dies auch so. Der moderne RPG-Programmierer vermeidet die Verwendung von Bezugszahlen und verwendet an ihrer Stelle die IF-, DOW-, DO-, DOU-, CASxx- und SELECT/WHEN-Operationen zum Steuern des Programmflusses.

Die meisten Operationscodes erzeugen einen entsprechenden Maschinenbefehl. Ein MOVE erzeugt einen Kopierbefehl, ein ADD erzeugt einen Addierbefehl, ein COMP erzeugt einen Vergleichsbefehl, ein IF erzeugt einen Vergleichsbefehl und so weiter.

Unterfelder (d. h. Feldgruppen und Datenstrukturen mit Mehrfachvorkommen) erzeugen zusätzlichen Overhead. Wenn dasselbe Feldgruppenelement in einer Routine verwendet wird, wobei das Feldgruppenelement in ein Feld verschoben wird und dann die Operationen in diesem Feld durchgeführt werden, dann kann dies die Laufzeit verbessern.

Manche Operationen, wie zum Beispiel die SQRT-, LOOKUP- und die dynamischen CALL-Operationen, erzeugen Unterroutinen, die dutzende von Maschinenbefehlen enthalten. Wenn zum Beispiel ein Programm die dynamische CALL-Operation verwendet, um ein externes Programm an acht verschiedenen Orten aufzurufen, wird die CALL-Operation in das Programm an allen dieser acht Stellen eingefügt. Der Kompiler erstellt denselben CALL-Befehl - achtmal. Dies kann die Programmgröße erhöhen.

Die dynamische CALL-Operation kann in eine Unterroutine plaziert werden und mittels der EXSR (führe Unterroutine aus) oder CASxx (vergleiche und führe Unterroutine aus)-Operationen durchgeführt werden. Die Unterroutine würde dann jedesmal aufgerufen, wenn das Unterprogramm benötigt wird. Der zum Aufrufen des Unterprogramms benötigte Code wird nicht an mehreren Orten verwendet, so dass das Programm kleiner ist und weniger Möglichkeiten für Programmierfehler liefert.

Der Overhead für den Aufruf einer RPG-Unterroutine ist minimal. Der Overhead für einen Anruf eines externen Programmes ist enorm Die Programmmodularität muss jedoch in Betracht gezogen werden. Die Verwendung einer Unterprozedur kann ein guter Kompromiss sein, genauso wie es ein gebundener Anruf (mittels der CALLB-Operation) zu einem anderen Programm sein kann, sowohl die Größe als auch die Leistung betreffend.

Andere zu betrachtende Punkte für die Programmgeschwindigkeit sind die Verwendung von MULT (Multipliziere) und DIV (Dividiere)-Operationen. Auf den meisten Computern sind diese Befehle Durchgangsbefehle; das bedeutet, dass sie ein leichtes „Aufstoßen“ der Maschine verursachen wenn sie durchgeführt werden. So laufen zum Beispiel die MOVE-, ADD- und SUB-Operationen um vielfaches schneller ab wie die MULT-, DIV-

und SQRT-Operationen. Die Anzahl, wie häufig ein Operationscode per Transaktion durchgeführt wird, muss auch in Betracht gezogen werden. Wenn der Code nur ein bis zwei mal pro Transaktion durchgeführt wird, dann muss man sich über die Geschwindigkeit keine Sorgen machen.

Da neue Kompilerarchitekturen in die RPG-Sprache eingeführt wurden, wird die Frage der Optimierung in bestimmten Gebieten vermindert. Die Kompiler oder eher Optimierer werden immer intelligenter.

Während sie immer klüger werden, wird mehr und mehr an Optimierung, die traditionell vom Programmierer durchgeführt wurde, nun vom Optimierer übernommen.

Bezugszahl-gesteuerte Logik

RPGII-Programmierer haben traditionell Bezugszahlen verwendet, um die Programmlogik zu steuern; lese einen Satz, schalte eine Bezugszahl an; vergleiche zwei Werte, schalte die Bezugszahl an; suche eine Feldgruppe, schalte eine Bezugszahl ein; beende das Programm, schalte eine Bezugszahl an.

Bezugszahlen waren die konsequenteste Art, die Programmlogik zu steuern. Eine Bezugszahl nimmt nur zwei Stellen in einer Kodierbestimmung ein und da der Festformat-Aufbau von RPG die Anzahl von Stellen einer Programmzeile begrenzt war, waren die Bezugszahlen eine logische Lösung.

Heutzutage jedoch wurde RPG verbessert, um strukturierte Operationen ähnlich der in PL/I zu unterstützen. Neue Operationen, wie zum Beispiel IF-THEN-ELSE-END, DO, DOW, DOU, SELECT-WHEN-OTHERWISE, versorgen RPG mit voll strukturierter Programmierunterstützung.

Wenn eine Bezugszahl verwendet wird, erstellt RPG einen Vergleiche-und-verzweige-Befehl. Wenn mehrere Bezugszahlen einen Operationscode steuern, werden mehrfache Vergleiche-und-verzweige-Befehle erzeugt. Dieser Effekt wurde in RPGII minimiert, indem die Bezugszahlen umgekehrt wurden und verwendet wurden, um eine GOTO-Operation zu steuern, die um den Programmcode herumspringt. Bild 83 und Bild 84 veranschaulichen diese ”RPGII-Art” des Kodierens in der RPG IV-Sprache.

```
.....CSRn01Factor1+++++++OpCode(ex)Factor2+++++++Result++++++++Len++DcHiLoEq
     C     FIELDA        COMP      '01'                                 22
     C  N22              GOTO      NOTOT1
     C                   ADD       QTY           TQTY
     C                   ADD       QTY           GQTY
     C                   MOVE      '**'          FLAG2
     C                   MOVE      '***'         FLAG3
     C     NOTOT1        TAG
```

Bild 83: Bezugszahl-gesteuertes Verzweigen

```
.....CSRn01Factor1+++++++OpCode(ex)Factor2+++++++Result++++++++Len++DcHiLoEq
     C     FIELDA        COMP      '01'                                 22
     C   22              ADD       QTY           TQTY
     C   22              ADD       QTY           GQTY
     C   22              MOVE      '**'          FLAG2
     C   22              MOVE      '***'         FLAG3
```

Bild 84: Traditionelle RPGII-Art der Bezugszahlverwendung

Bild 83 veranschaulicht die bevorzugte Methode zum Steuern der Programmlogik, bevor moderne Operationscodes verfügbar waren. Bild 84 veranschaulicht die Originalmethode des Steuerns der Programmlogik. *Keines der Verfahren repräsentiert jedoch eine brauchbare Kodiertechnik in der heutigen Welt fortschrittlicher Anwendungsprogrammierung.*

Wenn mehrere Bezugszahlen benötigt werden, um denselben Abschnitt eines Programms zu konditionieren, kann die OR-Be-

dingung (Spalten 7 und 8) verwendet werden - siehe Bild 85 und Bild 86.

```
.....CSRn01Factor1+++++++OpCode(ex)Factor2+++++++Result++++++++Len++DcHiLoEq
     C     FLDA          COMP       'A'                                21
     C     FLDB          COMP       'B'                                22
     C  N21
     CORN22              GOTO       NOTOT2
     C                   ADD        QTY            TQTY
     C                   ADD        QTY            GQTY
     C                   MOVE       '**'           FLAG2
     C                   MOVE       '***'          FLAG3
     C       NOTOT2      TAG
```

Bild 85: Mehrfach-Bezugszahl gesteuertes Verzweigen

```
.....CSRn01Factor1+++++++OpCode(ex)Factor2+++++++Result++++++++Len++DcHiLoEq
     C     FLDA          COMP       'A'                                21
     C     FLDB          COMP       'B'                                22
     C   21
     CAN 22              ADD        QTY            TQTY
     C   21
     CAN 22              ADD        QTY            GQTY
     C   21
     CAN 22              MOVE       '**'           FLAG2
     C   21
     CAN 22              MOVE       '***'          FLAG3
```

Bild 86: Traditionelle Multi-Bezugszahl-Steuerungslogik

Bild 85 veranschaulicht die bevorzugte Methode des Steuerns der Programmlogik mit mehreren Bezugszahlen, bevor der moderne Operationscode verfügbar war. Das Verfahren, das in Bild 86 gezeigt wird, wird leider häufiger angewendet. Dieser Kode ist tatsächlich von einer vorherigen RPG-Version übertragen worden. Um es noch einmal zu sagen: keines dieser Verfahren wird den heutigen Anforderungen des fortschrittlichen Anwendungsprogrammierens gerecht.