IF-opdragte

IF-opdrag en ooreenstemmende kode:

```
                    LDW  1,0,-4     x
                    LDW  2,0,-8     y
IF x < y            CMP  1,1,2
                    BGE  1,0,label
THEN StatSequence   code(StatSequence)
END                label:
```

Die spesiale instruksie `CMP` word gebruik om oorvloei te vermy. (In die interpreteerder werk `CMP` en `SUB` presies eenders omdat oorvloei ignoreer word.)

Verteenwoordiging van logiese waardes

Die volgende moet aangedui word:

- Watter register die berekende verskil bevat.
- Watter relasie (soos "=" of "<") verteenwoordig word.

Die rekord `Item` wat gebruik word om operande te verteenwoordig kry nou 'n nuwe veld `c` by om die relasie te verteenwoordig.

Voorwaardelike spronge

Die instruksies `BEQ`, `BNE`, ... `BGT` se opkodes volg op mekaar (40...45) en dit is handig om die relasie `c` voor te stel met ooreenstemmende kodes 0, 1, ... 5.

Prosedures `SimpleExpression` en `Relation` moet aangepas word om kode te genereer vir logiese uitdrukkings.

Prosedure `CJump` genereer die regte voorwaardelike sprong instruksie, maar die adresveld kan nie dadelik ingevul word nie. Dit word later ingevul deur 'n prosedure `Fixup` sodra die adres bepaal kan word.

Kodegenerasie: IF

```
...
ELSE sym = if THEN
  Get(sym); expression(x); CJump(x);
  IF sym = then THEN Get(sym)
  ELSE Error()
  END;
  StateSequence; Fixup(x.a);
  IF sym = end THEN Get(sym)
  ELSE Error()
  END
...
```

Voorwaardelike spronge

```
PROCEDURE CJump(VAR x: Item);
BEGIN
  IF x.type = Boolean THEN
    Put(BEQ+dual(x.c), x.r, 0, 0);
    ReturnRegister(x.r);
    x.a := pc-1
  ELSE Error()
  END
END CJump
```

Funksie `dual()` verskaf die duale bewerking wat ooreenstem met die kondisie. Byvoorbeeld as `c` gelik is aan 3 (betekenis >=) word 2 (<) verskaf. Die kode word dan gebruik om die regte instruksie te genereer.

Die adresveld bly tydelik 0 om later ingevul te word deur `Fixup`. Adresse is PC-relatief.

IF-THEN-ELSIF-ELSE

- Verskillende spronge word benodig.
- Voor elke `ELSIF` is 'n sprong na die einde van die struktuur—al hierdie instruksies se adresvelde moet later ingevul word. (Gebruik tydelik die adresvelde van die instruksies om die lys van instruksies voor te stel wat ingevul moet word.)

Voorbeeld: IF

```
IF i<j THEN i := 0
ELSIF i=j THEN i := 1
ELSE i := 2 i
END
```

```
   LDW  1,0,-4    i
   LDW  2,0,-8    j
   CMP  1,1,2
   BGE  1,0,3
   STW  0,0,-4
   BEQ  0,0,10
   LDW  1,0,-4
   LDW  2,0,-8
   CMP  1,1,2
   BNE  1,0,4
   ADDI 1,0,1
   STW  1,0,-4
   BEQ  0,0,3     unconditional jump
   ADDI 1,0,2
   STW  1,0,-4
   ... (first instruction following if)
```


WHILE

- Soortgelyk aan die IF-opdrag.
- Terugwaartse sprong aan einde van die lus.

```
WHILE i>0 DO i := i-1 END
```

```
   LDW  1,0,-4
   BLE  1,0,5
   LDW  1,0,-4
   SUBI 1,1,1
   STW  1,0,-4
   BEQ  0,0,-5   (jump back to start)
```

Algemene logiese uitdrukkings

Betekenis van EN en OF operatore:

```
p OR q = if p then TRUE else q
p & q  = if p then q else FALSE
```

Hierdie uitdrukkings vereis kode met 'n struktuur soortgelyk aan geneste if-opdragte:

```
IF (x <= y) & (y < z) THEN S END
```

word soortgelyk vertaal as

```
IF x <= y THEN
  IF y < z THEN S
  END
END
```


Voorbeeld: logiese uitdrukking

```
P = (a<b) & (c<d) & (e<f)
```

Die volgende struktuur is nodig:

```
   CMP  a,b
   BGE  false
   CMP  c,d
   BGE  false
   CMP  e,f
   BGE  false
true:
   ...
   ...
```

Die operator ~ (negering) word hanteer deur die duale kondisie te gebruik en die spronge om te ruil—geen ekstra kode is dus nodig nie.

Komplekse logiese uitdrukkings

Indien EN en OF operatore in komplekse patrone saamgevoeg word, word redelik komplekse lyste van spronge genereer.

```
Q = ((a<b) OR (c<d)) & ((e<f) OR (g<h))

   CMP  a,b
   BLT  true1
   CMP  c,d
   BGE  false
true1:
   CMP  e,f
   BLT  true2
   CMP  g,h
   BGE  false
true2:
```

Net soos by die vertaling van IF- en WHILE-opdragte is dit nodig om die adresvelde van voorwaardelike spronge later in te vul.