

## Capítol 6 Generació de Codi

6.1 Escriure la generació de codi en BNF pels següents operadors:

- (a) Divisió sencera ( $a \text{ div } b$ ) i mòdul ( $a \text{ mod } b$ )
- (b) Valor absolut ( $|n|$ )
- (c) Comparació múltiple de mes gran y mes petit ( $a < b < c$ ,  $a > b > c$ )

6.2 Escriure la generació de codi en BNF per les següents instruccions de control de flux:

- (a)  $\langle \text{instr\_dowhile} \rangle ::= \text{Do } \{ \langle \text{Instruccio} \rangle \} \text{ While } \langle \text{Expressio} \rangle$
  - (b)  $\langle \text{instr\_repeat} \rangle ::= \text{Repeat } \{ \langle \text{Instruccio} \rangle \} \text{ Until } \langle \text{Expressio} \rangle$
  - (c)  $\langle \text{i\_switch} \rangle ::= \text{Switch } \langle \text{Expressio} \rangle \text{ do}$ 
    - $\{ \text{Case } \langle \text{Constant} \rangle \text{ ":" } \langle \text{Instruccio} \rangle \}$
    - $[\text{Default}' : ' \langle \text{Instruccio} \rangle ]$
    - $\text{"end"}$
- $\langle \text{instr\_break} \rangle ::= \text{Break ;}$

## Capítol 6 Generació de Codi

6.3 Afegir a LS la possibilitat de construir records en les expressions. Exemple:

```
Type rec=record
  c1:integer;
  c2:character;
  c3:real;
end
...
var a:rec;
a=rec{ c1=>10,c3=>3.14};
```

6.4 Afegir a LS el switch al estil de C. Exemple:

```
switch a do
case 1:
  ...
  break;
case 2: ...
end
```

## Exercici 6.1

```
Rule <terme(cod,&t,Resultat)> ::= @Var t2;
  <factor(cod,t,Resultat)> {
    @if (!Resultat) throw Exception("No s'aprofita el resultat");
    @Var cod2=queue(); (
      * <factor(cod2,t2,Resultat)> ...
      | / <factor(cod2,t2,Resultat)> ...
      | mod <factor(cod2,t2,Resultat)>
        @switch ((t,t2)) {
          (TInt,TInt) => {}
          Others => throw Exception( "Error de tipus de dades en mòdul ",t,t2);
        };
        @t=TInt;
        @Cod2.Put(IModInt);
      | Div <factor(cod2,t2,Resultat)> ...
    )
    @cod.PutElements(cod2);
  }
```

## Exercici 6.1

```
Rule <terme(cod,&t,Resultat)> ::= @Var t2;
  <factor(cod,t,Resultat)> {
    @if (!Resultat) throw Exception("No s'aprofita el resultat");
    @Var cod2=queue(); (
      * <factor(cod2,t2,Resultat)> ...
      | / <factor(cod2,t2,Resultat)> ...
      | mod <factor(cod2,t2,Resultat)> ...
      | Div <factor(cod2,t2,Resultat)>
        @switch ((t,t2)) {
          (TInt,TInt) => {}
          Others => throw Exception(
            "Error de tipus de dades en divisió sencera ",t,t2);
        };
        @t=TInt;
        @Cod2.Put(IDivInt);
    )
    @cod.PutElements(cod2);
  }
```

## Exercici 6.1

```
Rule <factor(cod,&t,Resultat)> ::=
  @var v,nom,ets,nivell; (
  ! <factor(cod,t,Resultat)> ...
  | - <factor(cod,t,Resultat)> ...
  | "|" <Expressio(cod,t,Resultat)> "|"
    @if (!Resultat) throw Exception("No s'aprofita el resultat");
    @switch (t) {
      TInt => Cod.put(IAbsInt);
      TReal => Cod.put(IAbsReal);
      Others => throw Exception(
        "Error de tipus de dades a valor absolut ",t);
    };
  | ...
```

## Exercici 6.1

<b>p:</b>	<b>ILink 0</b>	<b>procedure p()</b>
	<b>IPushLit 4 10</b>	<b>begin</b>
	<b>IPushLit 4 2</b>	<b>    print 10 div 2, 10 mod 2, 3 , 3.14 ;</b>
	<b>IDivInt</b>	<b>end</b>
	<b>IPrintInt</b>	
	<b>IPushLit 4 10</b>	
	<b>IPushLit 4 2</b>	
	<b>IModInt</b>	
	<b>IPrintInt</b>	
	<b>IPushLit 4 3</b>	
	<b>IAbsInt</b>	
	<b>IPrintInt</b>	
	<b>IPushLit 8 3.14</b>	
	<b>IAbsReal</b>	
	<b>IPrintReal</b>	
	<b>IUnlink</b>	
	<b>IRet</b>	

## Exercici 6.1

**Rule** <FacBool(cod,&t,Resultat)>::=

@var t2,cod2=unbound,Comp;

@var EtiFalse=unbound, EtiTmpVar=unbound, sz;

<ExpArit(cod,t,Resultat)>

{

@if (!Resultat) throw Exception("No s'aprofita el resultat");

@if (cod2!=unbound) {

if (EtiFalse==Unbound) {

EtiFalse=Etiqueta("L\_CompFalse%");

EtiTmpVar=Etiqueta("L\_TmpVar%");

}

sz=MidaTipus(t2);

cod.Put(IStoreBVar,sz,RefEtiqueta(EtiTmpVar));

cod.Put(Comp);

cod.Put(IJumpFalse,RefEtiqueta(EtiFalse));

cod.Put(IPushBVar,sz,RefEtiqueta(EtiTmpVar));

t=t2;

};

## Exercici 6.1

```
@Var CompReal,CompInt; @cod2=queue();
(
  == @CompInt=IEqInt;      @CompReal=IEqReal;
  != @CompInt=INEqInt;    @CompReal=INEqReal;
  ">" @CompInt=IGreaterInt; @CompReal=IGreaterReal;
  "<" @CompInt=ILessInt;    @CompReal=ILessReal;
  >= @CompInt=IGreaterEqInt; @CompReal=IGreaterEqReal;
  <= @CompInt=ILessEqInt;   @CompReal=ILessEqReal;
) <ExpArit(cod2,t2,Resultat)>
  @switch ((t,t2)) {
    (TInt,TInt) => { Comp=CompInt; }
    (TInt,TReal) => { Cod.Put(IIntToReal); Comp=CompReal; }
    (TReal,TInt) => { Cod2.Put(IIntToReal); Comp=CompReal; t2=TReal; }
    (TReal,TReal) => { Comp=CompReal; }
    Others => throw Exception(
      "Error de tipus de dades en comparacio ",t,t2);
  };
  @t=TInt; @cod.PutElements(cod2);
}
```

## Exercici 6.1

```
@if (cod2!=unbound) {  
  cod.Put(Comp);  
  if (EtiFalse!=unbound) {  
    var EtiFin=Etiqueta("L_CompFin%");  
    cod.Put(IJmp,RefEtiqueta(EtiFin));  
    cod.Put(DefEtiqueta(EtiFalse));  
    cod.Put(IPushLit,4,0);  
    cod.Put(DefEtiqueta(EtiFin));  
    TS.VarSz=TS.VarSz+sz;  
    EtiTmpVar.Instanciar(-TS.VarSz);  
  }  
};
```

## Exercici 6.1

	p:	ILink 8		IPushLit 4 1
				IPushLit 4 2
procedure p() begin print 10<20; print 30<40<50; print 1<2<3<4; end		IPushLit 4 10		IStoreBVar 4 L_TmpVar8
		IPushLit 4 20		ILessInt
		ILessInt		IJumpFalse L_CompFalse7
		IPrintInt		IPushBVar 4 L_TmpVar8
				IPushLit 4 3
				IStoreBVar 4 L_TmpVar8
				ILessInt
				IJumpFalse L_CompFalse7
				IPushBVar 4 L_TmpVar8
				IPushLit 4 4
				ILessInt
				IJump L_CompFin9
				L_CompFalse7: IPushLit 4 0
				L_CompFin9: IPrintInt
				IUnlink
				IRet
			IPushLit 4 30	
		IPushLit 4 40		
		IStoreBVar 4 L_TmpVar4		
		ILessInt		
		IJumpFalse L_CompFalse3		
		IPushBVar 4 L_TmpVar4		
		IPushLit 4 50		
		ILessInt		
		IJump L_CompFin5		
		L_CompFalse3: IPushLit 4 0		
		L_CompFin5: IPrintInt		

## Exercici 6.3

Rule  $\langle \text{factor} \rangle ::= \dots$

| Identificador (...

| "{" [  $\langle \text{Camp} \rangle$  { ,  $\langle \text{Camp} \rangle$  } ] "}"

)

Rule  $\langle \text{Camp} \rangle ::= \text{identificador} \Rightarrow \langle \text{expressio} \rangle$

## Exercici 6.3

**Rule** <factor(cod,&t,Resultat)> ::= ...

| **Identificador** #(nom) (... |

@if (!(TypeP(ETSTipus,ets) && TypeP(TipusEstructura,Ets.Tipus)))

throw Exception(nom," no es u record");

@if (!Resultat) throw Exception("No s'aprofita el resultat");

@Var Camps=[(c,false, {

var cod=Queue();

cod.Put(IAddSP,MidaTipus(c.Tipus));

cod }) |c<-ets.Tipus.Camps];

"{" [ <Camp(Camps)> { , <Camp(Camps)> } ] }"

@for (c<-reverse(Camps)) Cod.PutElements(c[2]);

@t=ets.Tipus;

)

## Exercici 6.3

```
Rule <Camp(Camps)> ::= @var nom;  
    identificador #(nom)  
    @var Camp = Search (c <- Camps, c[0].nom == nom) c  
        else throw Exception(nom, " no es un camp");  
    @if (Camp[1]) throw Exception("Inicialització duplicada de ", nom);  
    @var CodCamp = Queue(), t;  
    => <expressio(CodCamp, t, true)>  
    @if (t != Camp[0].Tipus) throw Exception("Error de tipus");  
    @Camp[1] = true;  
    @Camp[2] = CodCamp;
```

## Exercici 6.3

	ICall main	procedure main()
	IExit	begin
main:	ILink 26	type St=record
	IPushLit 8 3.5	c1:integer;
	IPushLit 1 a	c2:character;
	IPushLit 4 10	c3:real;
	IPopBVar 13 -13	end;
	IAddSP 8	var a:St;
	IPushLit 1 a	a=St{C1=>10,C3=>3.5,C2=>'a'};
	IPushLit 4 5	var b:St;
	IPopBVar 13 -26	b=St{C1=>5,C2=>'a'};
	IUnlink	end
	IRet	

## Exercici 6.4

Rule <instruccio>::=

...

```
| switch <expressio> do  
  { <cassimple> : { <instruccio> } }  
  ( default : { <instruccio> } | $ )  
  "end"
```

```
| break ";"
```

Rule <cassimple>::= case ( [ - ] numero | caracter )

## Exercici 6.4

```
Rule <instruccio(cod,EtiEndSwitch)> ::= ...
| @var tsel,codBody=Queue();
  switch <expressio(cod,tsel,true)> do
    @if (tsel!:=TInt && tsel!:=TReal && tsel!:=TChar)
      throw Exception("Error de tipus");
    @TS.VarSz=TS.VarSz+MidaTipus(tsel);
    @var SelDesp= -TS.VarSz;
    @cod.Put(IPopBVar,MidaTipus(tSel),SelDesp);
    @Var EtiEndSwitch=Etiqueta("L_FiSwitch%");
    {
      @Var EtiCase=Etiqueta("L_Case%");
      @cod.Put(IPushBVar,MidaTipus(tSel),SelDesp);
      case <cassimple(cod,tsel)> :
        @Cod.Put(IJmpTrue,RefEtiqueta(EtiCase));
        @CodBody.Put(DefEtiqueta(EtiCase));
        { <instruccio(CodBody,EtiEndSwitch)> }
    }
}
```

## Exercici 6.4

( **default** :

  @Var EtiDefault=Etiqueta("L\_Case%");

  @cod.Put(IJmp,RefEtiqueta(EtiDefault));

  @CodBody.Put(DefEtiqueta(EtiDefault));

  { <instruccio(codBody,EtiEndSwitch)> }

| \$

  @cod.Put(IJmp,RefEtiqueta(EtiEndSwitch));

)

  @cod.PutElements(CodBody);

  @cod.Put(DefEtiqueta(EtiEndSwitch));

**"end"**

| **break** ";"

  @if (EtiEndSwitch==unbound) throw Exception("break fuera de switch");

  @cod.Put(IJmp,RefEtiqueta(EtiEndSwitch));

## Exercici 6.4

```
Rule <cassimple(cod,tse)> ::= @var v,signo=false;
[ - @signo=true; ] numero#(v) @if (signo) v=-v;
@if (tse!:=TInt && tse!:=TReal) throw Exception("Error de tipus en cas");
@switch (tse) {
  TInt=> if (TypeP(Real,v)) {
    cod.Put(IIntToReal);
    cod.Put(IPushLit,8,v);
    cod.Put(IEqReal);
  }
  else {
    cod.Put(IPushLit,4,v);
    cod.Put(IEqInt);
  }
  TReal=> {
    cod.Put(IPushLit,8,Coerce(Real,v));
    cod.Put(IEqReal);
  }
};
```

## Exercici 6.4

```
| @var v;  
  caracter#(v) @if (tsel!:=TChar) throw Exception("Error de tipus en cas");  
  @cod.Put(IPushLit,1,v);  
  @cod.Put(IEqChar);
```

## Exercici 6.4

```
main:   ILink 8
        IPushBVar 4 -4
        IPopBVar 4 -8
        IPushBVar 4 -8
        IPushLit 4 1
        IEqInt
        IJumpTrue L_Case64
        IPushBVar 4 -8
        IPushLit 4 2
        IEqInt
        IJumpTrue L_Case66
        IJump L_Case68
```

```
L_Case64:  IPushLit 4 1
           IPrintInt
           IJump L_FiSwitch63
L_Case66:  IPushLit 4 2
           IPrintInt
           IJump L_FiSwitch63
L_Case68:  IPushLit 4 4
           IPrintInt
L_FiSwitch63: IUnlink
           IRet
```

```
procedure main()
begin
  var a:integer;
  switch a do
    case 1 : print 1; break;
    case 2 : print 2; break;
    default: print 4;
  end
end
```