CONCLUSION BIBLIOGRAPHY

APPENDICES

CONCLUSION

One of the main goals of the project was to gain a good experience in compiler methodology. This goal has been achieved ; we also developed principles and techniques in the run-time system design as well as in the static management and in their interface.

By implementing the language in its whole, we had to control a huge bulk of information and one may ask oneself if this is really worthwhile. To this question we answer that complexity is a problem in itself ; having succeeded to master it in a reliable way is quite an achievement ; this has been made possible by carefully choosing basic principles and applying them in a systematic and modular way.

Let us now try to evaluate the translation process on the basis of the following criteria ; *efficiency, security, portability* and *design effort.*

Beforehand, a number of general considerations must be made.

On the one hand, some of these criteria are many-sided, e.g. the efficiency must be split up into compile-time and run-time efficiency and both of these have two aspects : time and space. On the other hand, these criteria are conflicting :

- an increase in efficiency, security and portability must be paid by an increase in design effort.
- a higher level of security and portability generally decreases the efficiency.
- a higher level of run-time efficiency generally decreases the compile-time efficiency in space and/or time.
- a higher run-time efficiency in space generally causes a decrease in run-time efficiency in time and vice versa.

In the implementation described in this book the stress has been laid on run-time efficiency, security and portability while keeping a reasonable degree of compile-time efficiency. In a number of cases, some compromises have been made in order to keep the design effort inside reasonable limits. However the principle according to which "run-time efficiency of simple language features must not be affected by the presence of more intricate features" (Samelson-Bauer) has been constantly cared for.

Now, the evaluation proper can be formulated, it is based on the actual implementation on the X8.

(1) *Compile-time efficiency in time* fits into quite acceptable limits ; it amounts to

$$5 + 6x \text{ seconds}$$

where x is the length of the source program in pages (one page = 60 average lines) [4].

(2) *Compile-time efficiency in space :* the whole ALGOL 68 system including debugging facilities and initialized tables occupy 100K memory words of 27 bits, among which about 70K instructions. However the use of overlay techniques allows to run the compiler within a 32K words direct access memory.

(3) *Run-time efficiency in time :* comparisons with the ALGOL 60-X8 compiler show an average increase in efficiency of 30% in favour of the ALGOL 68-X8 compiler. This is not negligeable given the ALGOL 68 compiler has few restrictions, given its level of portability and the simplicity of the mechanism of local optimizations.

(4) *Run-time efficiency in space :* this efficiency is difficult to estimate, we have no comparative figures at our disposal. Comparing the length of the object program with the length of the source program is meaningless : one single assignation *(x:=y)* may give rise to a variable number of objects instructions depending on the mode of x and y, i.e. on the data structure being assigned.

(5) *Portability*

We distinguish two main aspects in portability :

- the language in which the compiler is written (a).
- the algorithm itself of compilation (b).

(a) The X8-compiler has been designed in an ALGOL 68-like language but hand-coded in assembly language. The language used for the design has been defined as our experience was growing ; as a consequence, the design needs some polishing before it can be accepted by a compiler.

(b) The second aspect, i.e. the portability of the algorithm itself of compilation (in particular the code generator) has been solved quite satisfactorily in the X8 compiler :

- the interface with the operating system is pretty well localized and can be easily modified.
- up to the production of intermediate code, only routines of lexical analysis dealing with internal representation of numbers (and possibly strings) have to be reconsidered to be transferred on a new hardware.
- the machine code generation itself, as it should be clear from this book, is surprisingly portable even on computers with a minimal hardware (III.5.1). For register allocation, only declarations making the correspondence between formal and actual registers must be specified according to each particular hardware.

  Moreover, only a few routines have been made machine dependent in order to take direct advantage of particular hardware facilities.

  These routines are :

CONVERTACCESS	(III.1.3)
INREGPS	(III.1.2)
DEREFPS	(III.1.2)
GMI	(III.2.2)
COPYCELLS	(III.5.4.2)

Also *GTAB*, (section III.2.2) containing instructions to be generated in an interpretative form, and its interpretation routine should be rewritten. Finally, the global optimizer and the loader should probably be modified.

(6) *Design effort*

Roughly speaking, the compiler has consumed 20 men-years, but what does it really mean?

- The reader will be aware of how far we have optimized, and how few restrictions we have introduced.
- The programming tools we had at our disposal were very poor : the X8 assembler.
- The hardware and in particular the memory at our disposal was underdimensioned. As an example we had no protected backing store to save the compiler ; this means that it had to be reintroduced from cards and paper tape at each set of corrections or additions!
- The majority of the members of the team were unexperienced when they entered the project.
- In the 20 men years, the time spent for learning the language is incorporated, and we started reading early versions of drafts. Also, time spent for programming the special purpose system supporting the compiler and all debugging facilities is incorporated in the 20 men years.
- Finally, we must add that much time has been spent in checking the compiler carefully, step by step ; only a very small number of easily locatable bugs have been discovered since the compiler has been operational. It must be stressed that during the debugging process, the existence of high-level (design) and low-level (programming) documentation, carefully kept up to date, has appeared to be of the utmost importance.

BIBLIOGRAPHY

[1] A. van Wijngaarden (ed), B. J. Mailloux, J. E. L. Peck, C. H. A. Koster, *Report on the algorithmic language ALGOL 68*, MR 101, Mathematisch Centrum, February 1969.

[2] K. Samelson, F. Bauer, *Sequential formula translation*, Comm. ACM, February 1960.

[3] B. Randell and L. J. Russell, *ALGOL 60 implementation*, Academic Press, 1964.

[4] Kruseman - Aretz, *Het object programma gegenereerd door de X8-ALGOL-60 vertaler van het MC*, MR 121, Mathematisch Centrum, Amsterdam, Feb. 71.

[5] D. Gries, *Compiler construction for digital computers*, Wiley, 1971.

[6] D. E. Knuth, *Semantics of context-free languages*, Mathematical systems theory, vol. 2, n° 1, 1968.

[7] E. Irons, *A Syntax Directed Compiler for ALGOL 60*, Comm. ACM, January 1961.

[8] A. van Wijngaarden et al., *Draft revised report on ALGOL 68*.

[9] J. E. L. Peck (Editor), *ALGOL 68 Implementation*, North Holland Publishing Company, 1971.

[10] G. Schorr, W. Waite, *An efficient machine-independent procedure for garbage collection in various data structures*, CACM, August 67.

[11] P. Branquart, J. P. Cardinael, J. P. Delescaille, J. Lewi, M. Vanbegin, *Output of the syntactic analyzer of the ALGOL 68-X8.1 compiler*, Technical Note N73, MBLE Res. Lab., Part I (June 1971), Part II (Dec. 1971), and P. Branquart, J. P. Cardinael, J. P. Delescaille, J. Lewi, M. Vanbegin, *User's manual of the ALGOL 68-X8.1 system*, June 1973.

[12] P. Branquart, J. Lewi, *A scheme of storage allocation and garbage collection for ALGOL 68*, Report R133, MBLE Res. Lab., April 1970 and *ALGOL 68 Implementation*, J. E. L. Peck (Editor), North Holland Publishing Company, 1971.

[13] P. Branquart, J. Lewi, J. P. Cardinael, *Local generators and the ALGOL 68 working stack*, Technical Note N62, MBLE Res. Lab., Sept. 1970.

[14] P. Branquart, J. Lewi, *On the implementation of local names in ALGOL 68*, Report R121, MBLE Res. Lab., Sept. 70 and *Proceedings of the International Computing Symposium*, Bonn 1970.

[15] P. Branquart, J. Lewi, *On the implementation of coercions in ALGOL 68*, Proceedings of the International Computing Symposium, Bonn 1970, and MBLE Res. Lab., Report R123 .

[16] W. M. Mc Keeman, *Peephole optimization*, Comm. ACM, July 1965.

[17] P. Branquart, J. Lewi, M. Sintzoff, P. Wodon, *The composition of semantics in ALGOL 68*, Report R125, MBLE Res. Lab., Feb. 1970 ; CACM, Nov. 1971.

[18] P. Branquart, J. Lewi and J. P. Cardinael, *Analysis of the parenthesis structure of ALGOL 68*, Report R130, MBLE Res. Lab., April 1970 and *ALGOL 68 Implementation*, J. E. L. Peck (Editor), North Holland Publishing Company, 1971.

[19] P. Branquart, J. P. Cardinael, J. Lewi, *An optimized translation process, application to ALGOL 68*, R224, MBLE Res. Lab., and ICS Davos 1973.

[20] P. Branquart, J. P. Cardinael, J. P. Delescaille, J. Lewi, M. Vanbegin, *Data structure handling in ALGOL 68 compilation*, Proceedings of ALGOL 68 III International Conference, Winnipeg, June 1974 and MBLE Res. Lab. Report R254.

[21] P. Branquart, J. P. Cardinael, J. Lewi, *An optimized translation process and its application to ALGOL 68*, Report R204, MBLE Res. Lab., Part I, September 1972 ; Part II, January 1974, Part III, February 1974, Part IV, May 1974.

[22] J. P. Delescaille and F. Heymans, *On keeping the EL-X8 alive ; emulation on the BS*. Technical Note N101, MBLE Res. Lab., October 1975.

APPENDIX 1 : ANOTHER SOLUTION FOR CONTROLLING THE WOST% GARBAGE COLLECTION INFORMATION.

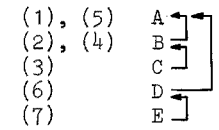
The idea of the solution is not ours, but it seems to be originated from the ALGOL 68-R compiler implemented on the ICL series by I. Currie. The solution is very efficient as such, however, it will be shown how, combined with our system, it would give still better results. It is to be noted that in such a combination practically the whole *gc* management described in this book remains valid.

The solution consists in constructing at compile-time a table representative of all possible *SWOST%* contents ; let us call it *GCTAB*. Each table element consists of the garbage collection information for one *WOST%* value (mode and access for example). Moreover elements are linked by a chain field in such a way a pointer to a table element gives access (through the chain field) to all the elements representative of a *WOST%* contents at a given moment.

Suppose for example the *WOST%* contents varies as follows :

(1) A
(2) A B
(3) A B C
(4) A B
(5) A
(6) A D
(7) A D E

The corresponding chain and entry points are scketched like this :



With this table, instead of having a dynamic *GCWOST%*, the garbage collection information for each *BLOCK%* reduces to a pointer *gcw%* stored in its *H%*, pointer to a *GCTAB* element. In principle, instructions are generated to update the *gcw%* of the current *BLOCK%* each time a value is stored or deleted from *WOST%* ; the corresponding static management requires a field on *BOST (gc1)* representative of the *GCTAB* entry point associated with the value.

Two optimizations are now possible :

(1) The first one corresponds to remark 2 at the end of I.2.4.3 : *gcw%* must **not** be updated at run-time if no garbage collection may take place during the time the corresponding value stays on *WOST%*. We can easily keep track of this fact by means of two new fields on *BLOCKTAB*, *gcp* and *gci* which are both pointers to *GCTAB*.

-*gcp* represents at each moment of the generation the actual *WOST%* state of the *BLOCK%*.

-*gci* is a static image of *gcw%* of that *BLOCK%*. Code updating *gcw%* will only be produced :

(a) when code risking to activate the garbage collector is produced, and

(b) when *gcp* and *gci* of the corresponding block are different.

(2) The second one corresponds to the minimization of the garbage collection information as explained in I.2.4.3.b, at the exception that what is minimized is the contents of *GCTAB* instead of *GCWOST%*. Clearly, the optimization is less advantageous here ; however some run-time actions may be saved given *gcp* will have to change less often and hence, instructions updating *gcw%* will have to be generated less frequently.

Practically, three compile-time routines allow to take care of the management of the *WOST%* garbage collection information :

(1) *PROTECT* will check whether a new value to be stored on *WOST%* has to be protected through *GCTAB* ; formulas given in II remain valid to make up that decision. When the value has to be protected through *GCTAB*, *gcl* in *BOST* and *gcp* in *BLOCKTAB* have to be updated.

(2) *DELETE* will cancel the protection of a given value deleted from *WOST%* by updating *gcp* in *BLOCKTAB*.

(3) *UPDATE* will generate *gcw%* updating code if it appears that on *BLOCKTAB* $gci \neq gcp$. This action is only to be taken when an ICI risking to call the garbage collector is to be generated. In *BLOCKTAB*, *gci* is updated accordingly.

APPENDIX 2 : SUMMARY OF THE SYNTAX

1. LBLOCK → lblockV BLOCKBODY

2.1 IDEDEC → idedecV FDECLARER iden = ACPAR
OPDEC → opdecV FDECLARER oper = ACPAR

2.2 LOCVARDEC → locvardecV ADECLARER variable |
locvardecV ADECLARER variable := SOURCE

2.3 HEAPVARDEC → heapvardecV ADECLARER variable
HEAPVARDEC → heapvardecV ADECLARER variable := SOURCE

3. LOCGEN → locV ADECLARER
HEAPGEN → heapV ADECLARER

4. LABELDEC → labeldecV label :
GØTO → gotoV label

5. CALL → callV PRIMCALL (ACPAR1, ACPAR2, ..., ACPARn)
FORMULA → dformulaV operator OPERAND1 OPERAND2 |
mformulaV operator OPERAND
ROUTDEN → routdenV (FORPAR1, FORPAR2,... FORPARn) : ROUTBODY
FORPARi → FDECLARER fideni

6. DEPROC → deprocV DEPROCCOERCEND
PROC → procV ROUTBODY

7. JPROC → jprocV label

8. CALLMODIND → callmodindV modind
MODEDEC → modedecV modind = ADECLARER

9. TRANSFORMAT → transformatV FORMATCOERCEND
FORMAT → formatV DYNREP

11.1 SELECTION → selectionV selector of SECONDARYSEL

11.2 DEREF → derefV DEREFCOERCEND

11.3 SLICE → sliceV PRIMSLICE INDEXERS
INDEXERS → [INDEXER1 , ..., INDEXERn]
INDEXERi → TRIMMER |
INDEX

11.4 UNITED → unitingV UNCOERCEND

11.5 ROWING → rowingV ROWCOERCEND

12.1 ASSIGNATION → assignation∇ DESTINATION := SOURCE

12.2 IDREL → idrel∇ TERTL {:=: | :≠:}$_1^1$ TERTR

12.3 CONFREL → confrel∇ TERTL {::= | ::}$_1^1$ TERTR

13. STDCALL → stdcall∇ (ACPAR1, ACPAR2, ..., ACAPRn)

14.2 SERIAL → LBLOCK | NONBLOCK
LBLOCK → lblock∇ BLOCKBODY
BLOCKBODY → NONBLOCK
NONBLOCK → SNONBLOCK | BALNONBLOCK
SNONBLOCK → PRELUDE last∇ LABUNIT
PRELUDE → {{DECLA ; | UNITV ; }$_0^\infty$ DECLA ;}$_0^1$ {LABUNITVS}$_0^1$
BALNONBLOCK → {PRELUDE last∇ LABUNIT . LABELDEC}$_0^1$
{LABUNITVS last∇ LABUNIT . LABELDEC}$_0^\infty$
LABUNITVS last∇ LABUNIT
LABUNITV → {LABELDEC}$_0^\infty$ UNITV
LABUNIT → {LABELDEC}$_0^\infty$ UNIT
LABUNITVS → {LABUNITV ;}$_0^\infty$
DECLA → IDEDEC | LOCVARDEC | HEAPVARDEC | OPDEC | MODEDEC

14.3 CONDCL → if∇ SERIALB CHOICECL fi
CHOICECL → then1∇ SERIAL |
thef∇ SERIALB CHOICECL |
then2∇ SERIAL1 else∇ SERIAL2 |
then3∇ SERIAL elsf∇ SERIALB CHOICECL

14.4 CASECL → case∇ CASECHOICE in∇ UNIT1, ... UNITn {out∇ SERIAL}$_0^1$ esac
CASECHOICE → UNITC | CASECONF

14.5 CASECONF → caseconf∇ mode1∇ TERTL1, ... moden∇ TERTLn {:: | ::=}$_1^1$ TERTR

15.1 COLLVOID → collvoid∇ (UNITV1, ..., UNITVn)

15.2 COLLROW → collrow∇ (UNITD1, ..., UNITDn)

15.3 COLLSTR → collstr∇ (UNITD1, ..., UNITDn)

16.3 FORCL → for∇ {from∇ UNITF}$_0^1$ {by∇ UNITB}$_0^1$
{to∇ UNITT}$_0^1$ {foriden}$_0^1$ {while∇ SERIALW}$_0^1$
do∇ UNITD

16.4 TRCALL → trcall∇ TRPRIM (UNIT1, ..., UNITn)

APPENDIX 3 : SUMMARY OF TOPST PROPERTIES

The table below shows how the fields *flextop* and *Δmem* of a *TOPST* element are initialized when a new *TOPST* element is set up by the activation of ρ(πα) or *NEWACTION (action)*. Other *TOPST* fields are initialized to 0. The letter T means that the corresponding field in the new element is copied from the old top one.

π(α)/action	*flextop*	*Δmem*
BLOCKBODY	T	0
FDECLARER	*(stat 0)*	0
ACPARi	*(stat 1)*	0
OPERANDi	"	0
ADECLARER	*(stat 0)*	0
SOURCE	*(stat 1)*	0
PRIMCALL	*(stat 0)*	0
FORPARi	"	0
ROUTBODY	*(dyn bn)*	0
DEPROCCOERCEND	*(stat 0)*	0
FORMATCOERCEND	"	0
DYNREP	"	0
SECONDARYSEL	T	T
DEREFCOERCEND	*(stat 0)*	0
PRIMSLICE	"	T
TRIMMER	"	0
INDEX	"	0
UNCOERCEND	*(stat 1)*	T+*Δunion*
ROWCOERCEND	*(stat 1)*	T+*Δrow*
DESTINATION	T	T
SOURCE	*(stat 1)*	T
TERTL	*(stat 0)*	0
TERTR	"	0
UNITV{i}	"	0
SERIALB	"	0
UNITC	"	0
TERTLi	"	0
collrow∇	T	T
UNITDi	*(stat 1)*	0
collstr∇	T	T
UNITF	*(stat 0)*	0
UNITB	"	0
UNITT	"	0
UNITD	"	0
SERIALW	"	0
TRPRIM	"	0
UNITi	"	0

APPENDIX 4 : SUMMARY OF THE NOTATIONS

1. MEM%

RANST% *(ranstpm%)*
HEAP% *(heappm%)*
DISPLAY%
BLOCK% | SBLOCK%
DBLOCK%

	Device	Static size	Current static pointers
SBLOCK%	H%	*h*	
	SIDST%	*sidsz*	*sidc*
	DMRWOST%	*dmrsz*	*dmrc*
	GCWOST%	*gcsz*	*gcc*
	SWOST%	*swostsz*	*swostc*
DBLOCK%	DIDST+LGST%		
	DWOST%		

2. H%

stch%
dch%
wp%
bn%
gcid% gcidp%
gcbodyflag%
gcw% gchp%
gcsz%

lblocks and pblocks

result% swostp%
gcp%
dmrp%
flex%
prevflag%
retjump%

pblocks only

3. DYNAMIC VALUE REPRESENTATION

Name : *pointer%*
scope%

Rowname : *pointer%*
scope%
descr%

Descriptor : *offset%*
states%
iflag%
do%
{*li%*
ui%
di%}*

Union : *overhead%*
value%

Routine : *constabp%*
scope%

Format : *constabp%*
scope%

Tamrof : *offset%*
ndrep%
constabp%

4. BLOCKTAB (entry : *bnc*)

BLOCK%	Pseudo-BLOCK%$_a$	Routine BLOCK%$_b$
sidsz	$sidsz_a = sidsz_{b1}$	$sidsz_b = sidsz_{b1} + sidsz_{b2}$
dmrsz	$dmrsz_a$	$dmrsz_b$
gcsz	$gcsz_a$	$gcsz_b$
swostsz	$swostsz_a$	$swostsz_b$
gcid	$gcid_a$	$gcid_b$ {*gcbodyflag*}
bn	bn_a	bn_b

5. *ACCESS (cadd)*

Fundamental	Accessory
(constant v)	*(intct v)* *(boolct v)* *(bitsct v)* *(charct v)*
(directtab a)	*(routct a)* *(formatct a)* *(tamroftct a)*
(diriden bnc.sidc) *(variden bnc.sidc)* *(indiden bnc.sidc)* *(dirwost bnc.swostc)* *(dirwost' bnc.swostc)* *(indwost bnc.swostc)* *(nihil 0)*	
	(ddisplay bn) *(dirabs a)* *(dirgcw bnc.gcc)* *(dirdmrw bnc.dmrc)* *(varabs a)* *(varwost bnc.swostc)* *(i2iden bnc.sidc)* *(i2wost bnc.swostc)* *(label bnc.labnb)*

6. SYMBTAB

Identifiers (IDENTAB)

Static property	Fields	Form
mode		
cadd	*class* *add* *hadd* *tadd*	(constant *v*) (directtab *a*) (diriden *bnc.sidc*) (variden *bnc.sidc*) (label *bnc.labnb*)
scope	*insc* *outsc*	
flagdecl *flagused*		

Mode indications (INDTAB)

mode		
cadd		(label *bnc.lo*)

7. BOST

Static property	Fields	Form
mode		*dectabp*
cadd	*class* *add* *hadd* *tadd*	
smr	*hadd* *tadd*	*bnc.swostc*
dmr		*(stat bnc.swostc)* *(dyn bnc.dmrc)* *nil*
gc		*bnc.gcc* *nil*
or	*kindo*	*iden* *var* *gen* *nil*
	bno *derefo* *geno* {*flexo*} {*diago*}	
scope	*insc* *outsc*	
{*flexbot*}		
obprogp		

8. TOPST

Action	Fields	Form
flextop	*class* *spec*	*(stat 0)* *(stat 1)* *(dyn bn)*
Δmem *countbal* *countelem* *flagnextbal*		

9. CONSTAB

Routines

Non-standard	Standard	Jump
lo	{specific}	*lo*
bnsc		*bnsc*
$sidsz_b$		
$dmrsz_b$		
$gcsz_b$		
$swostsz_b$		
$gcid_b$		
flagstand {0}	*flagstand* {1}	
flagjump {0}		*flagjump* {1}

Formats

lo
ndrep
bnsc
formstringp

APPENDIX 5 : LIST OF INTERMEDIATE CODE INSTRUCTIONS

This appendix is a complete list of the ICI's. With each of them, a number is given between brackets ; this number is the page number where the definition of the ICI is found.

1	STWOST1	MODE	
		CADDS	
		CADDO	(105)
2	STWOST2	MODE	
		CADDS	
		CADDO	(105)
3	STWOST3	MODE	
		CADDS	
		CADDO	(105)
4	STWORD	CADDS	
		CADDO	(97)
5	STADD	CADDS	
		CADDO	(104)
6	STGCWOST	MODE	
		CADD	
		CADDGC	(94)
7	STDMRWOST	CADD	
		CADDDMR	(106)
8	STACPAR	MODE	
		CADDS	
		CADDO	(109)
9	STDYNWOST1	MODE	
		CADD	(197)
10	STDYNWOST2	MODE	
		CADD	(197)
11	STDYNWOST3	MODE	
		CADD	(197)
12	STSTATWOST	MODE	
		CADDS	
		CADDO	(105)
13	STGCNIL	CADDGC	(97)
14	PLUS	CADDS	
		CADDO	(178)
15	STNDESCRWOST	MODE	
		CADD	(184)

16	STGCELEM	BNCS	
		GCCS	
		BNCO	
		GCCO	(254)
17	STOVERHUNION	MODE	
		CADD	(195)
18	STPLUS	CADD1	
		CADD2	
		CADDO	(177)
19	STWOSTINCR	MODE	
		CADDS	
		INCR	
		CADDO	(254)
20	STNAMEINCR	CADDS	
		INCR	
		CADDO	(179)
21	MINUS	CADDS	
		CADDO	(254)
22	STWP	BNC	
		CADD	(241)
23	STOREREG	MODE	
		CADD	(224)
24	LOADREG	MODE	
		CADD	(224)
25	INCRRTWOSTPM	CADDINCR	(199)
26	HOLE		(222)
27	JUMP	LABNB	(139)
28	LABDEF	LABNB	(139)
29	JUMPNO	LABNB	
		CADD	(214)
30	JUMPYES	LABNB	
		CADD	(254)
31	GOTO	BNC	
		BNCID	
		LABNBID	
		SWOSTC	(122)
32	LABFORMAT	CONSTABP	(254)
33	UPDCONSTAB	MODE	
		CONSTABP	(142)
34	CHECKSTAND	LABNB	
		CADD	(136)

```
35  LABID          LABNB                (121)

36  CHECKLAB       LABNB
                   CADD                 (148)

37  SWITCHCASE     LABNB
                   CADD1
                   CADD2                (230)

38  OBPROG         OBPROGP              (254)

39  BALTAB         BALTABP              (254)

40  INPROG                              (254)

41  OUTPROG                             (254)

42  INBLOCK        BNC                  (102)

43  CALLMIND       LRETURN
                   BNCRES
                   SWOSTCRES
                   LBODY                (162)

44  OUTMIND        BNBODY
                   N
                   CADD1
                   .....
                   CADDN                (163)

45  INMIND         BNCBODY              (163)

46  CALLDYNREP     LRETURN
                   BNCRES
                   SWOSTCRES
                   CADDFORMAT           (167)

47  INDYNREP       BNCBODY              (170)

48  OUTDYNREP      BNBODY
                   N
                   CADD1
                   .....
                   CADDN
                   FORMATSTRINGP        (171)

49  INITDYNREP     CADDFORMAT
                   CADDREP              (169)

50  INACPAR        BNCA
                   FLEX
                   CADDROUT
                   CADDRES
                   GCCRES
                   DMRCRES              (130)

51  CHECKDYNREP    LABNB
                   CADDFORMAT           (169)
```

```
52  CALL            LRETURN
                    CADDROUT
                    BNCA                (132)

53  RETURN          MODERES
                    CADDRES
                    BNBODY              (140)

54  STANDCALL       NPAR
                    DMRREC
                    CADDROUT
                    CADD1
                    *****
                    CADDN
                    CADDRES
                    DMRRES
                    GCRES
                    FLEX                (217)

55  STANDCALL1      LRETURN
                    N
                    BNCA
                    CADDROUT
                    CADD1
                    *****
                    CADDN               (137)

56  DEPROC          LRETURN
                    CADDROUT
                    CADDRES
                    GCCRES
                    DMRCRES
                    FLEX                (146)

57  STANDDEPROC     LRETURN
                    CADDROUT
                    CADDRES
                    GCCRES
                    DMRCRES
                    FLEX                (150)

58  CALLLAB         BNC
                    CADDROUT            (157)

59  STDCALLINOUT    CADDROUT
                    N
                    MODE1
                    CADD1
                    *****
                    MODEN
                    CADDN               (253)

60  CHECKFORMAL     MODE
                    CADD
                    N
                    CADD1
                    *****
                    CADDN               (110)

61  CHECKFLEX       CADD                (186)
```

62	CHECKFLEXR	BNCROUT	
		CADD	(186)
63	TRIMMER	NXDIM	
		CADDS	
		CADDL	
		CADDU	
		CADDL'	
		CADDOFF	
		CADDT	(187)
64	INDEX	NXDIM	
		CADDS	
		CADDI	
		CADDOFF	(188)
65	STINTERSTFL	CADDFLAG	
		CADDESCR	(254)
66	STFILLSTRIDE	MODE	
		CADDESCR	(188)
67	STNAME	CADDPOINTER	
		CADDSCOPE	
		CADDO	(191)
68	FILLSTATEONE	CADDESCR	(254)
69	STSCOPE	CADDS	
		CADDO	(254)
70	ROWINGSCADES	MODEO	
		CADDS	
		CADDO	(202)
71	ROWINGVAR	MODEO	
		CADDS	
		CADDO	(203)
72	ROWINGSCAL1	MODEO	
		CADDS	
		CADDO	(204)
73	ROWINGSCAL2	MODEO	
		CADDS	
		CADDO	
		DMRCS	(203)
74	ROWINGROW	MODES	
		MODEO	
		CADDS	
		CADDO	(200)
75	ROWINGREFSCA	MODEO	
		CADDS	
		CADDO	(206)

```
76  ROWINGREFROW   MODES
                   MODEO
                   CADDS
                   CADDO          (205)

77  ROWINGEMPTY    MODE
                   CADD           (175)

78  CHECKBOUNDS    CADDL
                   CADDU
                   CADDT          (240)

79  STBOUNDS       CADDL
                   CADDU
                   CADDT          (239)

80  STOVERHDESCR   MODEO
                   STATES
                   CADDD
                   CADDO          (239)

81  STFIRSTCOLLR   MODES
                   MODEO
                   CADDS
                   CADDO          (240)

82  STNEXTCOLLR    MODES
                   MODEO
                   CADDS
                   CADDO          (241)

83  STLITERALROW   MODE
                   CADDS
                   CADDO          (202)

84  ROWS           MODEPAR2
                   CADDROUT
                   CADDPAR1
                   CADDPAR2
                   CADDRES        (254)

85  ASSIGN         MODE
                   CADDS
                   CADDD          (209)

86  ASSIGNSCOPE    MODE
                   CADDS
                   CADDD          (209)

87  LOCVARGEN      MODE
                   CADD
                   N
                   CADD1
                   .....
                   CADDN          (112)
```

```
88  HEAPVARGEN      MODE
                    CADD
                    N
                    CADD1
                    .....
                    CADDN              (115)

89  FORTO           LABNB
                    CADDFORI
                    CADDBY
                    CADDTO             (250)

90  CHECKSCBLOCK    MODE
                    CADD
                    BN                 (104)

91  IDREL=          MODESL
                    CADDSL
                    CADDSR
                    CADDO              (211)

92  IDREL|=         MODESL
                    CADDSL
                    CADDSR
                    CADDO              (211)

93  LOCGEN          MODE
                    CADD
                    CADDGC
                    N
                    CADD1
                    .....
                    CADDN              (118)

94  HEAPGEN         MODE
                    CADD
                    CADDGC
                    N
                    CADD1
                    .....
                    CADDN              (119)

95  CONFTO          MODEL
                    MODER
                    CADDR
                    CADDO              (213)

96  CONFTOREC       MODEL
                    MODER
                    CADDO
                    CADDR
                    GCR
                    DMRR
                    CADD*R
                    GC*R
                    DMR*R              (213)
```

97	WIDEN	MODES	
		MODEO	
		CADDS	
		CADDO	(248)
98	STNIL	CADD	(174)
99	STSKIP	MODE	
		CADD	(173)
100	CONJWOST	CADD	(215)
101	CONJ	CADDS	
		CADDO	(215)
102	NOOPTIMIZE		(94)
103	NEWCARD	CARDNB	(254)
104	PRID	IDEN	(254)
105	PRNUMB	NUMB	(254)
106	CHECKNIL	CADD	(179)
107	CHECKOVERLAP	MODE	
		CADDS	
		CADDO	
		LABNB	(209)
108	STSTATACPAR	MODE	
		CADDS	
		CADDO	(109)
109	DEPROC1	LRETURN	
		CADDROUT	
		CADDRES	
		GCCRES	
		DMRCRES	
		FLEX	(152)
110	INBODY	BNCBODY	(154)

APPENDIX 6 : AN EXAMPLE OF COMPILATION

This example has been chosen very simple, it is only intended to give a flavour on how the successive stages of the compilation look like. (More examples can be found in [21]). The following comments on the output from the computer are useful :

(1) The source program text is first printed, its lines are numbered. The numbering, referred to as card number, is used in the next outputs as a reference to the source program.

(2) *DECTAB§* and

(3) *IDENTAB§* are self explaining. Note that the numbering in the first column is used for cross referencing.

(4) *BLOCKTAB§* must be explained :

line 25100 : corresponds to the block 'program'.

line 25103 : corresponds to 'particular program' with

-*bn=1* lexicographical depth number

-*sidsz=2* resulting from the way *Q(5)* is translated : the copy of the value of the primary *Q* is forced on *WOST%*, moreover, space is foreseen for the result of the call.

-*head* and *tail* are *IDENTAB§* pointers, keeping track of the declarations of the block, chained together ; they play the role of *gcid*.

line 25106 : corresponds to the routine possessed by *P* with

-*bn=1*, i.e. the scope 0 of the routine +1.

-*sidsz=2* corresponding to parameter *X* and variable *A* (blocks are merged).

-*swostsz=1*, foreseen for the result of *10+X*.

line 25109 : corresponds to the pseudo-block of the actual parameter of the call *Q(5)*, with

-*bn=2*, lexicographical depth number.

-*sidsz=1* for the integer parameter.

(5) Follows the linear prefixed form of the program i.e. *SOPROG* for the IC generation. This form is self-explaining. Note however that

- prefix markers start with '$',
- the lines **** CARD refer to the source program (1)
- the numbers 235, 236,... respectively refer to *IDENTAB§* entries 30584, 30588,...
- the number 232 refers to the operator *op(int,int)int* + in the initialized part of *INDTAB§* not printed here.
- coercions are kept in a separate table *COERCTAB* ; connections with *SOPROG*

are obtained through the specification field of the prefix markers of the coercends ; here : $ID and $DEN.

(6) The intermediate code (*OBPROG§*) should be easily understood in the light of PART II of this book. Only some details may differ from what has been described. Note moreover that *CONSTAB§* referred to in some ICI's is not printed here.

(7) The machine code in its relocatable form is then printed ; in this code, we find :
(a) instruction lines consisting of :
- the opcode using mnemonics. Note that A, S, G and F are registers and that SUBC means a subroutine call. The opcode may be preceded by one letter U, Y or N to mean a conditional execution of the instruction.
- an optional star the presence of which means (hardware) literal addressing.
- STAT, MPQ, MA and MS which are addressing types :
-STAT means normal addressing.
-MPQ means (hardware) display addressing.
-MA (MS) means indexed addressing using A (S) register.
-the address field consisting of a pair of integers, the first one being only significant in case of display addressing. This field may be followed by P, Z or E, which causes the setting up of conditions, subsequently to the execution of the instruction.
-finally the field symbolic which is a symbolic representation of run-time routines or static working cells. In case it is 'LABTABPI' however, it has a special meaning, indicating to the loader that address conversion using *LABTAB§* is involved.

(b) loader commands :
-LABDEF for label definition.
-LABROUT and OFFSECT issued from the IC *updconstab* with mode parameters *proc* and *string* respectively.

(c) lines with '****' :
They correspond to references to the ICI's currently translated.

(8) The loader automatically prints the starting addresses of :
- the object program *OBPROG%*,
- *RANST%* and
- *HEAP%*

(9) The actual result of the program execution is finally printed.

1. SOURCE PROGRAM

```
1        (
2         PROC P=(INT X)INT :(INT A:=10+X;
3                                        A));
4        PR 51PR  PR 62PR   PR 63PR   PR 64PR
5         PROC (INT  )INT Q:=P;
6         PRINT(("RESULT=",Q(5)))
7        )
```

2. DECTAB

```
15500    PROC
             MODERES        INT
             NMBPAR          1
             PARAM1         INT
15503    PROC
             MODERES        INT
             NMBPAR          1
             PARAM1         INT
15506    REF                15503
```

3. IDENTAB

	cadd			*used*	*mode*	*scope*		*chain*	*length*	*alpha*
30584	ROUTCT	0	1342	1	15500	0	0	0	1	P
30588	DIRIDEN	2	0	1	INT	0	0	30592	1	X
30592	VARIDEN	2	1	1	REF INT	1	1	0	1	A
30596	VARIDEN	1	0	1	15506	1	1	0	1	Q

4. BLOCKTAB

	swostsz	*sidsz*	*bn*	*head*	*gosz*	*tail*	*dmrsz*
25100	0	0	0	0	0	0	0
25103	3	2	1	30596	0	30596	0
25106	1	2	1	30588	0	30592	0
25109	0	1	2	0	0	0	0

5. *SOPROG*

0	****CARD	1
1	****CARD	2
2	(CLO	2
3	<RAN	0
4	$CONSDE	0
5	DECLAR	15500
6	DEFID	235
7	(R	0
8	SCOPE	0
9	DECLAR	15500
10	(F	0
11	DECLAR	14570
12	DEFID	236
13	)F	0
14	DECLAR	14570
15	:	0
16	(CLO	1
17	<RAN	0
18	$LVARDE	0
19	DECLAR	14570
20	DEFID	237
21	:==	0
22	$FORMUL	0
23	DYADOPE	232
24	$DEN	0
25	SINTCT	10
26	SEP	0
27	$ID	0
28	ID	236
29	;	0
30	$LASTUN	0
31	****CARD	3
32	$ID	1
33	ID	237
34	>NAR	0
35	)OLC	1
36	)R	0
37	;	0
38	$LVARDE	0
39	DECLAR	15503
40	****CARD	4
41	PRAGNUMB	62
42	PRAGNUMB	63
43	****CARD	5
44	PRAGNUMB	64
45	DEFID	238
46	:==	0
47	$ID	0
48	ID	235
49	;	0
50	$LASTUN	0
51	$CALL	0
52	****CARD	6
53	$ID	0
54	ID	152
55	(AP	0
56	(COLL	1
57	COLLMOD	14555
58	$DEN	5
59	STRINCT	0
60	,	0
61	$CALL	7
62	$ID	3
63	ID	238
64	(AP	0
65	$DEN	0
66	SINTCT	5
67	)PA	0
68	)LLOC	1
69	)PA	0
70	****CARD	7
71	>NAR	0
72	)OLC	2

COERCTAB

0	END	0	0
1	DEREF	0	14587
2	END	0	0
3	DEREF	0	15506
4	END	0	0
5	[]OUTTYPE	8	14660
6	END	0	0
7	[]OUTTYPE	8	14570

6. OBPROG$

```
  0          INPROG

  1      ****CARD1

  2      ****CARD2

  3          INBLOCK        BNC                          1

  4          JUMP           LABNB                        3

  5      C2:
  6          LOCVARGEN      MODE              INT
                            CADD       DIRIDEN      2    1
                            N                            0

  9          STANDCALL      LRETURN                      0
                            NLONG                        0
                            N                            2
                            BNC                          2
                            DMRC       NIL*
                            CADDROUT   ROUTCT       0  1129
                            CADD 1     INTCT        0    10
                            CADD 2     DIRIDEN      2     0
                            CADDRES    DIRWOST      2     0
                            DMRCRES    NIL*
                            GCCRES                       0
                            FLEX       STAT              1

 17          ASSIGN         MODES             INT
                            CADDS      DIRWOST      2     0
                            CADDO      VARIDEN      2     1

 20      ****CARD3

 21          HOLE

 22          UPDCONSTAB     MODE                        15500
                            CADD       ROUTCT       0  1342

 24          RETURN         MODERES           INT
                            CADDRES    DIRIDEN      2     1
                            BNBODY                       1

 27      C3:
 28       ****CARD4

 29          PRAGMAT           62

 30          PRAGMAT           63

 31      ****CARD5

 32          PRAGMAT           64

 33          LOCVARGEN      MODE                        15503
```

```
                             CADD         DIRIDEN      1       0
                             N                                 0

 36          ASSIGN          MODES                         15500
                             CADDS        ROUTCT       0    1342
                             CADDO        VARIDEN      1       0

 39      ****CARD6

 40          UPDCONSTAB      MODE                          14660
                             CADD         DIRCTTAB     0    1345

 42          STWOST3         MODE                          15503
                             CADDS        DIRIDEN      1       0
                             CADDO        DIRWOST      1       0

 45          INACPAR         BNCACPA                           3
                             FLEX         STAT                 0
                             CADDROUT     DIRIDEN      1       0
                             CADDRES      DIRWOST      1       2
                             GCCRSE                            0
                             DMRCRES                           0

 49          STACPAR         MODE                  INT
                             CADDS        INTCT        0       5
                             CADDO        DIRIDEN      3       0

 52          CHECKSTAND      LABNB                             4
                             CADD         DIRWOST      1       0

 54          CALL            LRETURN                           5
                             CADDROUT     DIRWOST      1       0
                             BNCACPA                           3

 57      C4:
 58          STANDCALL1      LRETURN                           5
                             N                                 1
                             BNC                               3
                             CADDROUT     DIRWOST      1       0
                             CADD 1       DIRIDEN      3       0

 62      C5:
 63      ****CARD7

 64          STDCALLINOUT    LRETURN                           0
                             N                                 2
                             BNC                               1
                             DMRC         NIL*
                             CADDROUT     ROUTCT       0     565
                             MODE 1                        14660
                             CADD 1       DIRCTTAB     0    1345
                             MODE 2                INT
                             CADD 2       DIRWOST      1       2
                             CADDRES      NIHIL        0       0
                             DMRCRES      NIL*
                             GCCRES                        NILGC
                             FLEX         STAT                 0

 75          HOLE

 76          STADD           CADDS        DDISPLAY     0       1
```

```
                               CADDO      DIRABS     0     20
79          STWORD             CADDS      INTCT      0      0
                               CADDO      DIRABS     0     17
82          STWORD             CADDS      VARABS     0   4095
                               CADDO      DDISPLAY   0      1
85       ****CARD8
86      L1:
87          OUTPROG
```

7. *MACHINE CODE*

```
        opcode     lit  addtype   addr          symb
  0      LDA        *   STAT      0     11
  1      LDS        *   STAT      0      0
  2      LDG        *   STAT      0     11
  3      STG            STAT      0      0      INCRGC9
  4      LDG        *   STAT      0      0
  5      SUBC       *   STAT      0      0      INPROG9
         ****                            1
         ****                            2
         ****                            3
  6      LDG        *   STAT      0      2
  7      STG            STAT      0      0      CARDNB
  8      LDA        *   STAT      0      1
  9      LDS        *   STAT      0      0      GCINFOTAB
 10      LDG        *   STAT      0     16
 11      SUBC       *   STAT      0      0      INBLOCK19
 12      LDA        *   STAT      0     13
 13      LDS        *   STAT      0      0
 14      LDG        *   STAT      0      2
 15      SUBC       *   STAT      0      0      INBLOCK29
         ****                            4
         ****                            5
 16      GOTO       *   STAT      0     29      LABTABPI
         ****                            6
         ****                            9
         LABDEF                          2
 17      LDS        *   STAT      0     10
 18      ADS            MPQ       1     11
         ****                           17
         ****                           20
         ****                           21
         ****                           22
 19      STS            MPQ       1     12
         LABROUT                        17      LABTABPI
         ****                           24
 20      LDG            MPQ       1      4
 21      STG            STAT      0      0      RET.JUMP
 22      LDS            MPQ       1      5
 23      LDA        *   MPQ       1     12
 24      LDG        *   STAT      0      1
 25      SUBC       *   STAT      0      0      UPDATEDISP
 26      LDG            MA        0      0
 27      STG            MS        0      0
 28      GOTO           STAT      0      0      RET.JUMP
```

```
                ****                                27
                ****                                28
                ****                                29
                ****                                30
                ****                                31
                ****                                32
                ****                                33
                ****                                36
                LABDEF                               3
29              LDG         *   STAT      0      1342      CONSTABPI
30              STG             MPQ       1        11
31              LDG             STAT      0         0      DISPLAYPI
                ****                                39
                ****                                40
32              STG             MPQ       1        12
                OFFSECT                           1345
                ****                                42
33              LDF             MPQ       1        11
                ****                                45
34              STF             MPQ       1        13
35              LDG         *   STAT      0         6
36              STG             STAT      0         0      CARDNB
37              LDG         *   STAT      0      1358      CONSTABPI
38              LDS             MPQ       1        11
39              SUBC        *   STAT      0         0      INACPARDYN
                ****                                49
40              LDS         *   STAT      0         5
                ****                                52
41              STS             MPQ       2        11
42              LDA             MPQ       1        13   Z
43    Y         LDS         *   STAT      0        14
44    Y         GOTO        *   STAT      0         0      ALARM
45    U         LDA             MA        0         0   P
                ****                                54
46    N         GOTO        *   STAT      0        53      LABTABPI
47              LDA         *   STAT      0        57      LABTABPI
48              STA             STAT      0         0      RET.JUMP
49              LDA         *   STAT      0         1
50              LDS         *   MPQ       1        13
51              LDG         *   STAT      0         2
52              GOTO        *   STAT      0         0      CALLDYN
                ****                                57
                ****                                58
                LABDEF                               4
53              LDG         *   STAT      0         6
54              STG             STAT      0         0      CARDNB
55              LDS         *   STAT      0        21
56              GOTO        *   STAT      0         0      ALARM
                ****                                62
                ****                                63
                ****                                64
                LABDEF                               5
57              SUBC        *   STAT      0         0      SAVETIME
58              LDS         *   STAT      0      1345      CONSTABPI
59              LDA             MS        0         2   Z
60    Y         GOTO        *   STAT      0        70      LABTABPI
61              LDA         *   STAT      0         0      VALSTPI
62              LDG         *   STAT      0         1
63              STG             STAT      0         2      VALSTPI
64              LDG         *   STAT      0      1000
65              SUBC        *   STAT      0         0      INITLOOPR9
                LABDEF                               6
66              SUBC        *   STAT      0         0      PRINTCHARS
```

```
67      LDG       *  STAT     0      66      LABTABPI
68      LDA       *  STAT     0       0      VALSTPI
69      SUBC      *  STAT     0       0      FINALLOOPR9
        LABDEF                              7
70      LDS       *  MPQ      1      15
71      SUBC      *  STAT     0       0      PRINTINTS
72      SUBC      *  STAT     0       0      RESTTIME
        ****                               75
        ****                               76
73      LDA          STAT     0       1      DISPLAYPI
74      SBA       *  STAT     0     256
        ****                               79
75      STA          STAT     0       0      RTWOSTPM
76      LDS       *  STAT     0       0
        ****                               82
77      STS          STAT     0       0      RTBNA
78      LDS       *  STAT     0       0      NIL
        ****                               85
        ****                               86
79      STS          STAT     0       1      DISPLAYPI
        ****                               87
        LABDEF                              1
80      GOTO      *  STAT     0       0      FINAL9
```

8. LOADER INDICATIONS

```
OBPROGPI  11368
STACKPI   11449
HEAPPI    29700
```

9. PROGRAM RESULT

```
RESULT=         15
```