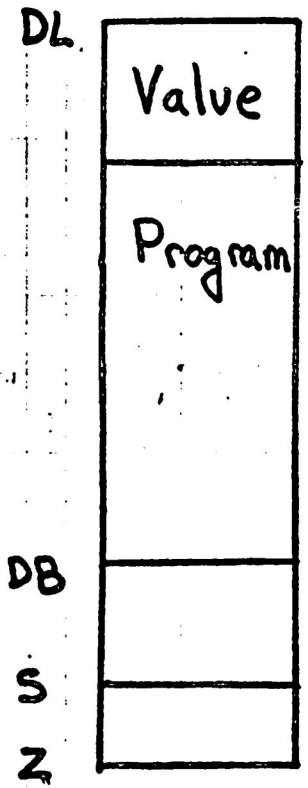
BASIC COMPILER - TERRY HAMM

BASIC Interpreter
Runtime Environment

24 code segments ~37000 words

>RUN PROG



PROG Data Area

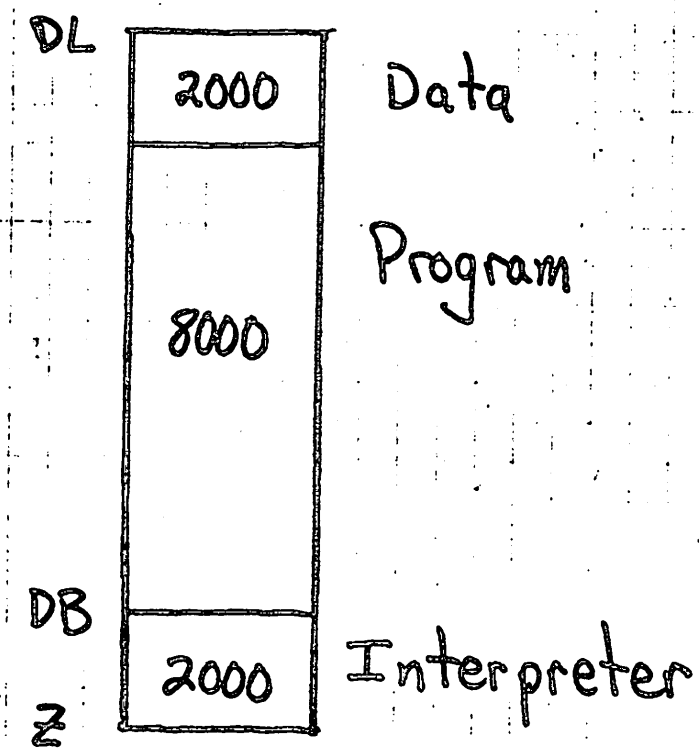
Program Area
- Symbol Table
- Statements
- Label Table

Data Area
Runtime Stack } Interpreter

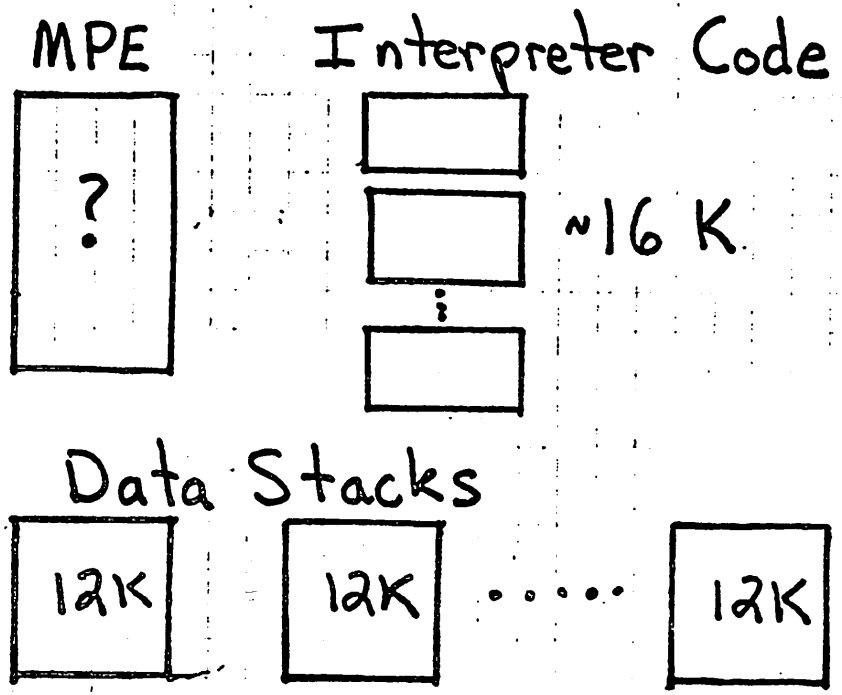
Data - 2000 words
Arrays, variables
file buffers

Program - 8000 words

Interpreter - 2000 words



Total
12K

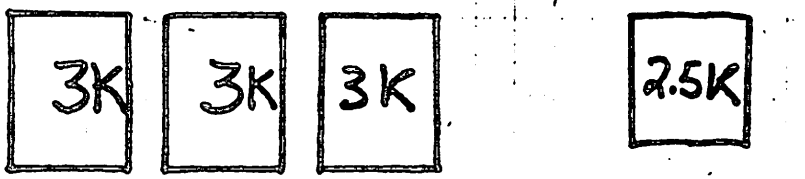


n	Memory
2	40 K
4	64 K
8	112 K
12	160 K
16	208 K

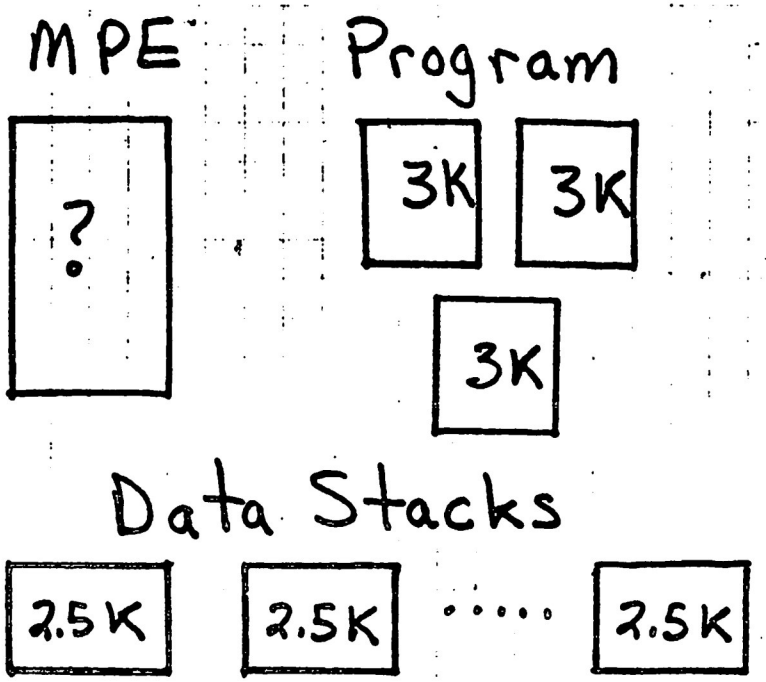
$12 \cdot n + 16$

FORTRAN

Program 9000 words
3 segments
Data 2500 words



Total 11.5K



n	Memory
2	14 K
4	19 K
8	29 K
12	39 K
16	49 K

$2.5 \cdot n + 9$

$\sim \frac{1}{4}$

HP 3000 BASIC COMPILER

BASIC Compiler

- Extension of Interpreter
- Compatible to Interpreter
- Runs as subsystem under MPE
- Commands control the compiling

Objectives

- Performance - improve BASIC program execution
  1. Sharable Code
  2. Smaller data stacks
  3. Executable code
- Compatibility
- Generality

User Interface

:BASICOMP [commandfile][,[uslfile][,listfile]]

	default file	use
command file	$STDIN	Input subsystem commands
uslfile	$OLDPASS	Output of RBM's
listfile	$STDLIST	List output

:BASICPREP [commandfile][,[progfile][,listfile]]

:BASICGO [commandfile][,listfile]

Subsystem Commands

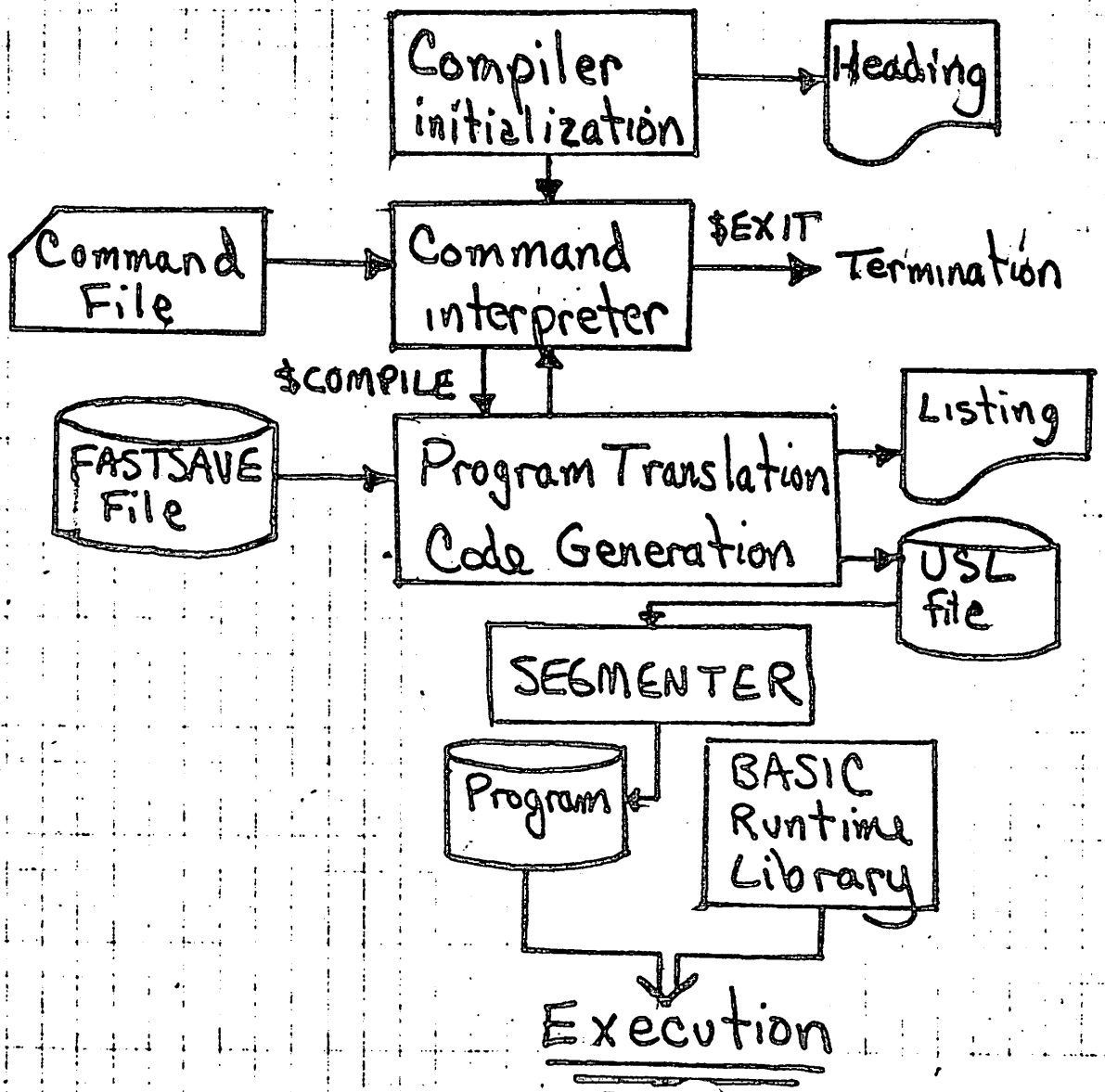
$CONTROL	parameter list
$ COMPILE	program name list
$ ENTRY	chain invoke list
$ TITLE	character string
$EXIT	

$CONTROL parameters

LIST	NOLIST
SOURCE	NOSOURCE
LABEL	NOLABEL
CODE	NOCODE
MAP	NOMAP
LINES =	number of lines
WARN	NOWARN
USLINIT	
SUBPROGRAM	START= name
SEGMENT =	name
INIT	

```
$CONTROL SOURCE,USLINIT,NOWARN
$CONTROL LABEL,LINES=50,START=C
$TITLE "PROGRAM C", <<MAIN>> "JAN 25"
$COMPILE  C(100,200)
$CONTROL  NOSOURCE,NOLABEL
$ COMPILE  A,B
$ ENTRY  A,B
$ ENTRY  SLPROG
$ EXIT
```

Compilation Process



13

```
$COMPILE  A, B, C
```

SEG'

B'RUNOB
A
B
C

```
$ CONTROL  SEGMENT=SEG1
$ COMPILE  A, B
$ CONTROL  SEGMENT=SEG2
$ COMPILE  C
```

SEG'

B'RUNOB

SEG1

A
B

SEG2

C

10

B'RUNOB

- Open files BASICIN - BASICOUT
- Allocate file buffers
- Enable arithmetic & library traps
- Initialize runtime global variables
- INVOKE to START program
- Handle termination upon return from START

BASIC runtime library

- COM area - file space management
- I/O utilities
  - Formatting
  - File manipulation
- String routines
- Matrix routines
- Builtin functions

Runtime Stack