Tech Info Library

Pascal III: Accessing the extra memory (1 of 5)

Revised: 11/30/84
Security: Everyone

Pascal III: Accessing the extra memory (1 of 5)

==========

Apple III Pascal is upwardly compatible from Apple II Pascal. One of the constraints this imposed on the design of Pascal III was the restriction of the data space to 64K bytes. This restriction has been made clear on all specifications of the product. However, since the system uses additional space for SOS, drivers, graphics, the interpreter and code segments, this restriction may still interfere with programs that use large quantities of data.

To illustrate the steps involved in accessing more data, below are some small assembly routines. They ask SOS to allocate more space for the program, allow transfers of data back and forth to that space, then deallocate the space. On top of that is built a simple string package which stores up to 128K of string values in this space. Though the memory within this space may be managed only in the simplest of ways, the space can still be very useful.

The routines consists of three portions:

A. An assembly language routine containing macros that the SOS interface needs.

B. A fancy version of MOVELEFT that moves bytes from one location in banked memory to another. It understands the memory addressing of the Apple III, and so it increments pages accordingly.

C. A Pascal unit that uses the first two programs to implement the string routines.

```
; Macro Definitions

.Macro Pull
; Pull 2 bytes off the stack and store them
PLA
STA %1
PLA
STA %1+1
```

```
.EndM

.Macro Push
; Load 2 bytes and put them on the stack
LDA %1+1
PHA
LDA %1
PHA
.EndM

.Macro Return
; Load the return address, PUSH it and RTS
Push %1
RTS
.EndM

.Macro EnterProc
; Save the return address
Pull %1
.EndM

.Macro EnterFunc
; Save the return address and kill the 4 byte bias
EnterProc %1
PLA
PLA
PLA
PLA
.ENDM

.Macro PushTrue
; Put a Boolean True on the stack
LDA #0
PHA
LDA #1
PHA
.ENDM

.Macro PushFalse
; Put a Boolean FALSE on the stack
LDA #0
PHA
PHA
.ENDM

.Macro P_A_Word
; Copies a word from a Pascal Var to assembly language
.If %1 & 0FF00 <> 0
WrongOrderInA_P_Word
.ENDC
LDY #0
LDA (%1),Y
```

```
STA %2
INY
LDA (%1),Y
STA %2+1
.ENDM

.Macro A_P_Word
; Copies a byte from a assembly lang word to a Pascal word
.IF %2 & 0FF00 <> 0
WrongOrderInA_P_Word
.ENDC
LDY #0
LDA %1
STA (%2),Y
INY
LDA %1+1
STA (%2),Y
.ENDM

.Macro P_A_Byte
; Copies a byte from a Pascal byte to assembly language
.If %1 & 0FF00 <> 0
WrongOrderInP_A_Byte
.EndC
LDY #0
LDA (%1),Y
STA %2
.ENDM

.Macro A_P_Byte
; Copies a byte from assembly language to a Pascal word
.If %2 & 0FF00 <> 0
WrongOrderInA_P_Byte
.ENDC
LDY #0
LDA %1
STA (%2),Y
TYA
INY
STA (%2),Y ; Clear highbyte of Pascal var
.ENDM

.Macro SOSCall
; Framework for calls to the SOS memory manager
BRK
.Byte %1
.Word %2
.ENDM

Temp1 .EQU 0E0
Temp2 .EQU 0E2
Temp3 .EQU 0E4
Temp4 .EQU 0E6
```

```
    ; Procedure Allocate(Var NumPages,Segnum,Bank,SegBase:
                        integer);external;
    ; {allocates a chunk of SOS memory:
    ;   Input:
    ;     NumPages: Maximum number of pages to try for.
    ;   Output:
    ;     NumPages: Number of pages actually allocated.
    ;     SegNum:   SOS Segment number (for deallocate)
    ;     Bank:     Starting address bank number
    ;     SegBase:  Starting address byte address ($0200..$9E00)}
```

Apple Tech Notes

Tech Info Library Article Number:639