

## Procedures and Functions in PeANUt

- number systems (bases) in .mli files
- procedure / function calls
- nested procedures
- the stack:
  - stack pointer register
  - stack addressing mode
  - the stack frame
- ref: [PeANUt Spec, ]; additional reading: [O'H&Bryant, sect 3.7]
- revise 2006 exam, Q2(a–d) (C Programming)

## Simple Procedure Calls

- motivation:
  - often, the instruction set does not include some operation that is regularly required
  - the user can effectively extend the instruction set by using procedures / functions
  - procedures can be written in PeANUt (like functions in C)
- PeANUt procedures:
  - the instructions CallProcedure and Return are important
  - CallProcedure allows the PC to be *remembered*, and a new PC value is given (so that execution can continue from a different place)
  - Return retrieves the *remembered* PC value and resets the PC to this value (so that execution continues where it left off)

## Using Other Bases in .mli files

- writing all instructions in binary can be tedious, although is often clearer
- notation:
 

<ul style="list-style-type: none"> <li>■ octal (o) (1 digit → 3 bits)                             <ul style="list-style-type: none"> <li>o100 → 001 000 000</li> <li>o123 → 001 010 011</li> <li>o767 → 111 110 111</li> </ul> </li> <li>■ hexadecimal (h) (1 digit → 4 bits)                             <ul style="list-style-type: none"> <li>h10 → 0001 0000</li> <li>h79 → 0111 1001</li> <li>h9D → 1001 1101</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>■ decimal (d) (n digits → 16 bits)                             <ul style="list-style-type: none"> <li>d5 → 0000 0000 0000 0101</li> <li>d64 → 0000 0000 0100 0000</li> <li>d79 → 0000 0000 0100 1111</li> </ul> </li> <li>■ address (a, octal) (n digits → 10 bits)                             <ul style="list-style-type: none"> <li>a5 → 0 000 000 101</li> <li>a17 → 0 000 001 111</li> <li>a167 → 0 001 110 111</li> </ul> </li> </ul>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
- examples:
  - 001 001 0 000 101 000 → o1 o1 a50 ; load 50\_8
  - 110101 0 000 000 011 → o6 o5 a3 ; trap 3
  - 1110101 000 000 000 → hE o5 a0 ; compXR

## Procedure Example: procedure-example.mli

- write a program that prints out \*A\*B\*

Print A

Print B

Executions of routine to print \*

```

START a10

AT a10
110100          a30      ; a10 call a30
o0 o1 0 001 000 001  ; a11 load 'A'
110101          a3      ; a12 trap 3 (put)
110100          a30      ; a13 call a30
o0 o1 0 001 000 010  ; a14 load 'B'
110101          a3      ; a15 trap 3 (put)
110100          a30      ; a16 call a30
110101          a1      ; a17 trap 1(halt)

AT a30
; procedure to print '*'
o0 o1 0 000 101 010  ; a30 load '*'
110101          a3      ; a31 trap 3 (put)
1110000        o000    ; a32 return
                
```





