Australian National University

Computer Science Courses
ANU College of Engineering & Computer Science

can we assume that the initial memory block for screen display is all zeros'

Week 09

Students are expected to attempt ALL of the questions in the Multi choice, Short answer and Lab quesitons. They will be discussed and marked at the beginning of the lab.

Multi choice

Question: Where does the stack pointer begin in rPeANUt? D	**A)** 0xFFF0 **B)** 0x7FFF **C)** 0x7001 **D)** 0x7000
Question: Which of the following is not a real store addressing mode? C	**A)** Base+displacement store **B)** Indirect store **C)** Immediate store **D)** Absolute store
Question: Recursion depth in rPeANUt is limited by: A E	**A)** Stack size **B)** Stack frame size **C)** rPeANUt's recursive depth limit **D)** All of the above **E)** A and B
Question: How do you use block to reserve 20 words of memory initialised to 0? D	**A)** `block 20 0` **B)** `block #20` **C)** `block 20=0` **D)** `block 20` **E)** `block #20=0`

difference's

Short answer

invalid call may cause infinite loop

Question: Why does the stack frame need to be planned carefully when doing a procedure call?
Question: How many words of memory could you fit into the stack before running off the end of the memory?

Lab questions

0x7000-0x7FFF= 4095

Excercise: In rPeANUt write a function (that use the standard rPeANUt approach for the stack frame) which squares the integer it is given. i.e. The c code would be:

```
int square(int n) {
   return n*n;
}
```

Test your function by implementing and running the square function on different values. What is the largest value you function can square?
Excercise: Write a program in rPeANUt that fills the graphics display with white and then halts. Using the command line count how many instructions it uses (compare with others in the lab). Also dump the frame buffer to a text file (using the -dump option) and check it is the same as whitescreen.dump. Use the "diff" command.
Excercise: Write a procedure which takes an x,y coordinate and sets a pixel on the screen to white. Use the following stack frame:

```
; setpixel - set a pixel to white
; stack frame :
;  return address #0
;  y #-1
;  x #-2
; "pixel (x,y) will be bit x%32 of the word at
```

```
; address 0x7C40 + 6*y + x/32" from spec
```

Test your approach with a program that set particular pixels.

- Include your setpixel method into the below program.

```
;  for (i = 0; i <= 100; i++) {
;    setpixel((rand())%192,(rand())%160);
;  }

0x0100 :   jump loopbool
loop:      push R0   ; work out x pixel
           call rand
           pop R0
           load #192 R1
           mod R0 R1 R0
           push R0

           push R0   ; work out y pixel
           call rand
           pop R0
           load #160 R1
           mod R0 R1 R0
           push R0
           call setpixel ; draw the pixel
           pop R0
           pop R0
           load loopcount R1
           add R1 ONE R1
           store R1 loopcount
loopbool : load #100 R0
           load loopcount R1
           sub R0 R1 R1
           jumpnz R1 loop
           halt

loopcount : block 1
mask : block #0x7fffffff ; we need this mask so the random number is positive

; rand - generate psudo random numbers.
; this uses Linear Congruential Generator see http://en.wikipedia.org/wiki/Linear_congruential_generato
; stack frame :
;   return address #0
;   return random value #-1
rand : load aval R0
       load rval R1
       mult R0 R1 R1
       load cval R0
       add R1 R0 R1
       store R1 rval
       rotate #12 R1 R1
       load mask R0
       and R0 R1 R1
       store R1 #-1 SP
       return
rval : block #0  ; this stores the current random
aval : block #1664525
cval : block #1013904223
```

Now when you run the program you should see a random cloud of pixels. Use the command line and the -dump option to check that what you have is the same as what is expected. cloud.dump

In-class Group Task

Write a recursive function in rPeANUt which calculates the greatest common divisor of two numbers. Here is the C implementation:

```
int gcd(int x, int y)
{
        // x >= y and y >= 0
        if (y == 0)
                return x;
        else
                return gcd(y, x % y);
}
```

UPDATED: **10 April 2013** / RESPONSIBLE OFFICER: **Head of School** / PAGE CONTACT: **COMP2300 Course Webmaster**