CS221 Assembly Language

Lab 06: Stacks and Procedures

Objectives

- Creating and using procedures
- Getting input from the user
- Using push and pop to store values
- Using registers to pass parameters to a procedure
- Using the stack to pass parameters to a procedure
- Using pushad, popad, pushfd, and popdf

Exercises

1. Create a procedure called "`find_odd_numbers`" (slide 19). In this procedure, ask the user to enter a number `n`. Call `ReadDec` to get an unsigned value from the user. `ReadDec` stores the entered value in the `eax` register. See Table 1 for an example and more information on `ReadDec`. This procedure should print the first `n`th odd numbers.
2. Create `arr_0` which is an array of `dwords` having 7 random elements.
    a. Create a procedure named "`reverse_array`" to reverse the elements in the array `arr_0`. Put the address of the array in `ebx` in the main procedure (slides 27 & 28) and store the size of the array in the `esi` register (slides 27 & 28). You must use loops, push, and pop (slides 8 – 11) to reverse the elements. Preserve `ebx` and `esi` using the `uses` directive (slide 31).
    b. Create a procedure "`print_array_r`". Pass to this procedure the address of `arr_0` in the `ebx` register and pass the size of `arr_0` to the array in `esi`. Preserve the value of `ebx` and `esi` using `push` (slide 22).
    c. Create a procedure "`print_array_s`". This procedure is similar to the procedure in the previous task except that you should pass the **address** and **size** of `arr_0` in the stack (slides 40 & 41).
3. Create a procedure called "`get_nth_factorial`". Ask the user for a decimal value `n` using `ReadDec`. Then, return the `n`th factorial in the `eax` register. Print the value from the `main` procedure.
4. Create a procedure named "`push_all`". This procedure should push all the registers and flags (slides 15 & 16) to the stack. Does this cause a problem when returning from the procedure? Can you solve it? (slide 25).

Table 1

Procedure	Registers	Example	Description
WriteDec	eax	mov eax, 10 call WriteDec	Prints the unsigned value in eax.
WriteInt	eax	mov eax, -8 call WriteInt	Print the signed value in eax.
Crlf	-	call Crlf	Prints a newline.
Clrscr	-	call Clrscr	Clears the screen.
WriteString	edx	.data msg byte "Hello, World!", 0 .code mov edx, offset msg call WriteString	Print a null-terminated string. edx has the address of the first character in a string.
WriteChar	al	mov al, "z" call WriteChar	Print the character in al.
ReadDec	eax	Call ReadDec ; Unsigned character will be in eax	Reads an unsigned value to eax. Will store 0 in eax if the input is invalid
ReadInt	eax	Call ReadInt ; Signed value will be in eax	Reads a signed value to eax. Will store 0 in eax if the input is invalid
ReadString	edx ecx eax	mov edx, offset buffer mov ecx, buffer_size call ReadString	Read a string from the user. Store the address of the buffer in edx. Store the size of the buffer in ecx. Get the number of read bytes in eax.
ReadChar	al	Call ReadChar	Read a character from the user and store it in al.