8. Advanced Procedures

Tuesday, October 22, 2024 11:39 AM

* How do you provide a procedure what it needs?
  * Passing arguments through registers
  * Arguments to a procedure can be pushed onto the stack before calling a procedure.

old way	new way
mov eax, num1	push num1 = 8
mov ebx, num2	push num2 = 10
call AddTwo	call AddTwo

* pass arguments by value or pass arguments by reference
  - ↳ Array or String
* pushing arguments onto the stack (cannot push a byte)
  * push word or dword only
* ESP - register that points to the last element added to the stack.

```
push num1 (dword)   ← 8
push num2 (dword)   ← 10
call AddTwo
```

↓ push, jump

	Offset	Stack	
	1000	8	[EBP+12]
stack pointer	996	10	[EBP+8]
	992	Return Address	[EBP+4]
ESP →	988	EBP	

↓ EBP ↗

* dec. or inc. with push or pop.
* will not directly modify ESP.

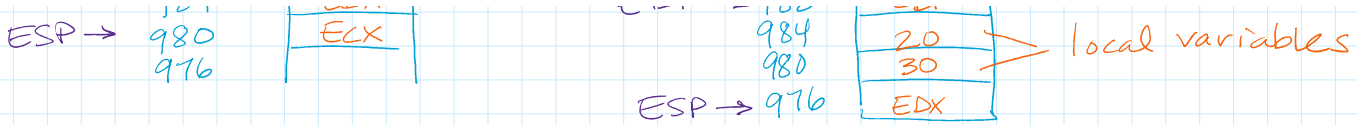
```
AddTwo proc
    push EBP          ← preserve its value
    mov EBP, ESP      ← make a copy of ESP (base stack pointer)

    mov EAX, [EBP + 12]   ; move 8 into EAX
    add EAX, [EBP + 8]    ; add 10 to EAX
    pop EBP           - restore EBP
    ret 8             → add 8 to ESP (clears the arguments of the stack)
                      → pop & jump.
AddTwo endp
```

* Justification for Base Stack Pointer (EBP)

	offset	Stack
+12	1000	8
+8	996	10
+4	992	ret. addr.
EBP →	988	EBP
	984	EDX
ESP →	980	ECX
	976	

offset	Stack	
1000	8	arguments
996	10	
992	ret. addr.	
EBP → 988	EBP	
984	20	local variables
980	30	




```
SomeProc  Proc
          push  EBP
          mov   EBP, ESP
          Sub   ESP, 8
          Push  EDX

          mov [EBP-4], 20    ; local var
          mov [EBP-8], 30    ; local var
          pop EDX            ; restore EDX
          mov ESP, EBP       ; removes local variables
          pop EBP

          ret 8
SomeProc  endp
```