Assembly

- funct proc uses ebx ecx
- funct proc [language type] uses register list [parameters]
- funct proc Near32
  push ebp
  mov ebp, esp
- funct proc stdcall uses ebx ecx esi,
  lpArray: dword, ; comment
  numValues: dword, ; comment
- funct(&array, Num)
- stdcall means that the function itself cleans up the stack, not the calling program
- f proc stdcall uses ... ...
  iVal1: dword,
  iVal2: dword
  local x: dword,
  y: dword,
  t: dword
- int f(sVal, dVal, bVal) →
  f proc stdcall uses ... ...,
  sVal: word,
  dVal: dword,
  bVal: byte
- INVOKE works only if you specify a language type
- INVOKE f, sX, dx, bVal1
- INVOKE f, 0, 4, 3
- g(&array, dVal)
  g PROC stdcall uses , dfarr: dword, dNum: dword
- INVOKE g, offset array, 27
- every parameter comes in as a 4 byte value
- INVOKE g, addr array, 27

- g PROTO stdcall, lpArray:ptr word, dVal:dword
- for a prototype, use everything but the uses directive
- list prototypes under .stack
- ExitProcess PROTO stdcall, dExitValue:dword
  INVOKE ExitProcess, 0
- f(a,b)
  {
  int [] ix = new int [5] sub esp, 20
- lea esi, [ebp-4] ; get address from EBP, subtract 4, then load into esi
-