Value: 4F52

Little Endian: [52 4F] (x86)

Big Endian: [4F 52] (RISC)

**X86**

Real vs Protected

* Real mode is the processor state when it is first powered on and only supports a 16-bit instruction set
* Protected mode is the processor state supporting virtual memory, paging, and other features; it is the state in which modern operating systems execute

The terms used to describe sizes in the x86 architecture are:

* byte: 8 bits
* word: 2 bytes
* dword: 4 bytes (stands for "double word")
* qword: 8 bytes (stands for "quad word")

**Ring Level**

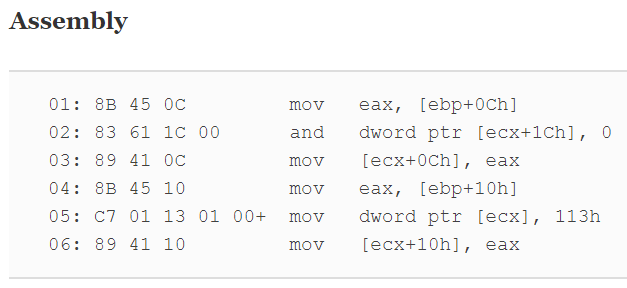
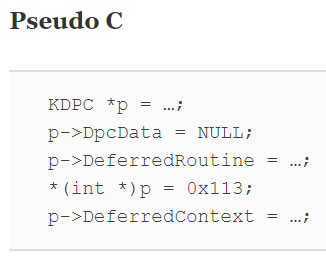
* 4 ring (0, 1, 2, 3)
  + 0 highest privilege to 3 lowest privilege

**Registers**

* x86 architecture has eight 32-bit general-purpose registers (GPRs):
  + EAX, EBX, ECX, EDX
  + ESI & EDI
  + EBP & ESP

|  |  |
| --- | --- |
| **Register** | **Purpose** |
| ECX | Counter in loops |
| ESI | Source in string/memory operations |
| EDI | Destination in string/memory operations |
| EBP | Base frame pointer |
| ESP | Stack pointer |

* The common data types are as follows:
* **Bytes**—8 bits. Examples: AL, BL, CL
* **Word**—16 bits. Examples: AX, BX, CX
* **Double word**—32 bits. Examples: EAX, EBX, ECX
* **Quad word**—64 bits
  + While x86 does not have 64-bit GPRs
  + It can combine two registers, usually EDX:EAX, and treat them as 64-bit values in some scenarios. For example, the RDTSC instruction writes a 64-bit value to EDX:EAX
* Instruction format: ***instr dest, src***
  + ecx = \*eax → mov ecx, [eax]
    - set ECX to the value at address EAX
  + \*eax = ebx → mov [eax], ebx
    - set the memory at address EAX to EBX
  + \*eax = 1 → mov dword ptr [eax], 1
    - Set the memory at address EAX to 1
  + \*(esi+34) = eax → mov [esi+34h], eax
    - set the memory address at (ESI+34) to EAX
  + eax = \*(esi+34) → mov eax, [esi+34h]
    - set EAX to the value at address (EAX+34)
  + edx = \*(ecx+eax) → mov edx, [ecx+eax]
    - set EDX to the value at address (ECX+EAX)
* C → Assembly Examples



* + Line 1 reads a value from memory and stores it in EAX
  + The DeferredRoutine field is set to this value in line 3
  + Line 2 clears the DpcData field by AND'ing it with 0
  + Line 4 reads another value from memory and stores it in EAX
  + The DeferredContext field is set to this value in line 6