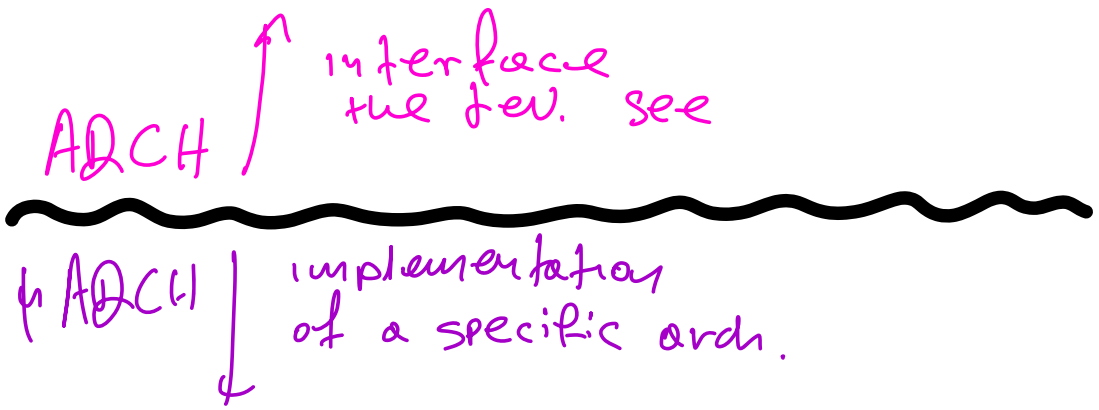
×86

x86

- 4004 → 1971 (4-bit)
- 8008 → 1972 (8-bit)
- 8086 → 1978 (16-bit)
  - Core i7-8086k (2018)
- 80186 → 1982 (16-bit)
- 80286 → 1982 (16-bit)
- 80386 → 1985 (32-bit)
- i486 → 1989 (32-bit)
- Pentium (P5) → 1993 (32-bit)
- Pentium II → 1997 (32-bit)
  - P6 (Pentium Pro) → 1995
- Pentium III → 1999 (32-bit)
- Pentium 4 → 2000 (32-bit)
  - Xeon, M, Celeron

...

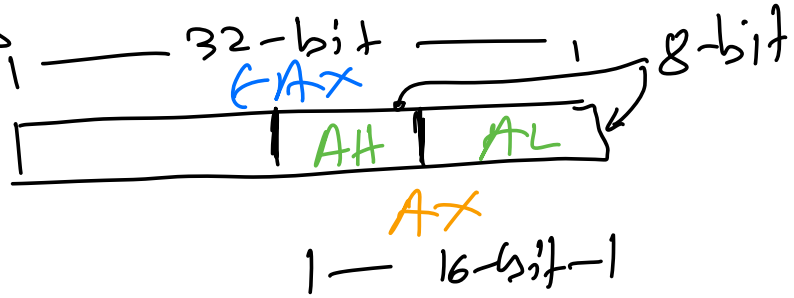
- Pentium D → 2005 (64-bit)
- Pentium Dual-Core → 2006 (64-bit)
- Core 2 → 2006 (64-bit)
- Core i3 → 2006 (64-bit)
- Core i5 → 2009 (64-bit)
- Core i7 → 2008 (64-bit)
- Core i3 → 2017 (64-bit)



x86 ISA (32-bit)

(Instruction Set Architecture)

- 8 General Purpose Registers
  - EAX
  - EBX
  - ECX
  - EDX
  - ESI
  - EDI
  - EBP
  - ESP



No (8-bit) "high", "low" parts!

- 2 Special Purpose Registers
  - EFLAGS → | | | ... | SF | ZF | | AF | | PF | | 1 | CF |
  - EIP

•) Instruction Set

⇒ Data Transfer Instructions
↳ mov, xchg, push, pop, ...

⇒ Arithmetic and Logic Instructions
↳ add, sub, ..., and, or, xor, not, ...

⇒ Control Instructions ⇒ change the value of EIP
↳ call, ret, jmp, JCC
↳ condition code (FLAGS)

⇒ Misc Instructions
↳ nop, enter, leave, test, cmp, ...
↳ int, syscall

[⇒ variable-length instructions
⇒ 0, 1, 2, 3, ... operands]

Examples

IA-32 ASM

AT&T syntax

- mov %eax, %ebx
  - SRC
  - DST
- add $0x123, %ecx
- push $0xdeadbeef
- pop %edx
- ...

"template"

INST	SRC	DST
	%r	%r
	$CONST	%r
	$CONST	[MEM]
	[MEM]	%r
	%r	[MEM]

• Memory Addressing

⇒ Direct

```
mov $0x1, 0x1110
mov 0x2000, %eax
```

⇒ Indirect

```
mov $0x1, (%eax)
mov %ebx, 0x10(%eax)
mov 0x10(%eax,%ebx,{1,2,4}), %edx
```

base register (BR)

offset (OFF)

index register (IR)

scale

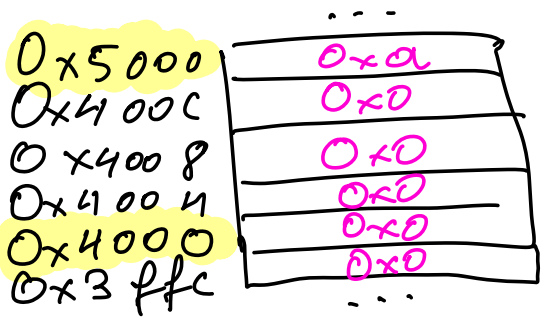
$$EA = BR + IR * \{1,3,4\} + OFF$$

Effective Address

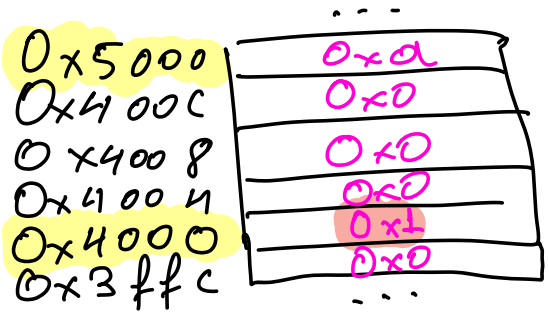
%eax = 0x1

mov $0x1, 0x4000

mov 0x5000, %eax



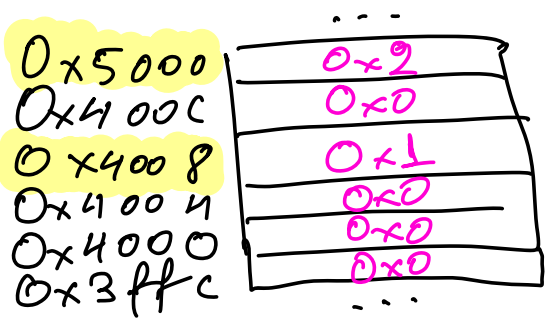
%eax = 0xa



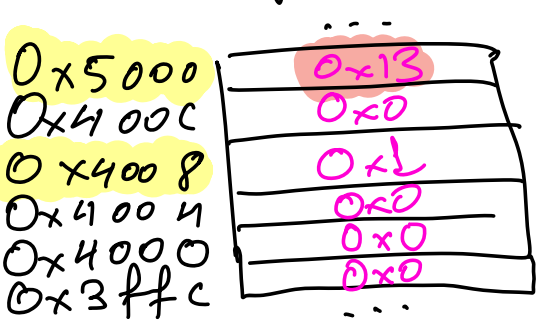
%ebx = 0x10

add 0x4008, %ebx

add %ebx, 0x5000



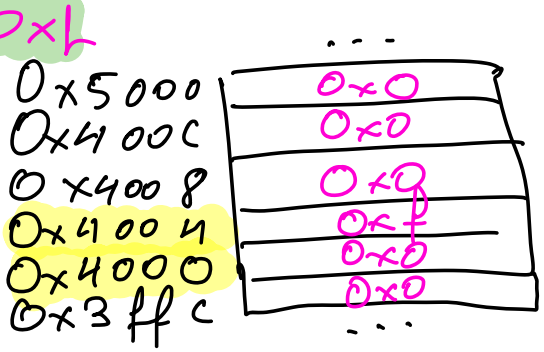
%ebx = 0x11



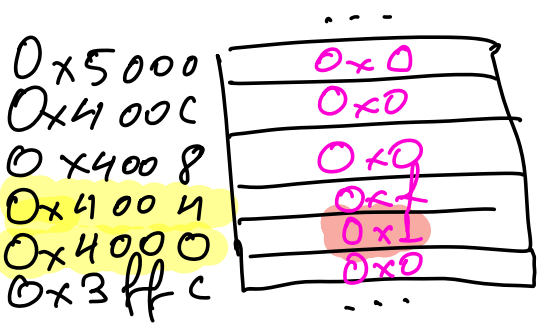
%eax = 0x4000, %ebx = 0x1

mov $0x1, (%eax)

mov 0x4(%eax), %ebx



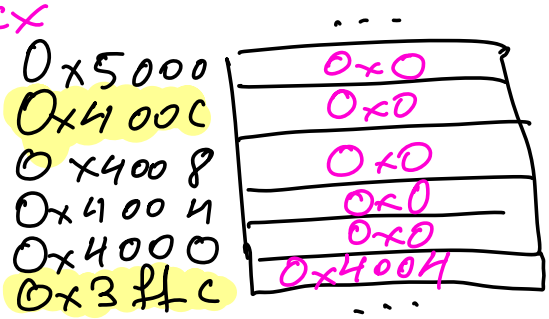
%ebx = 0x1



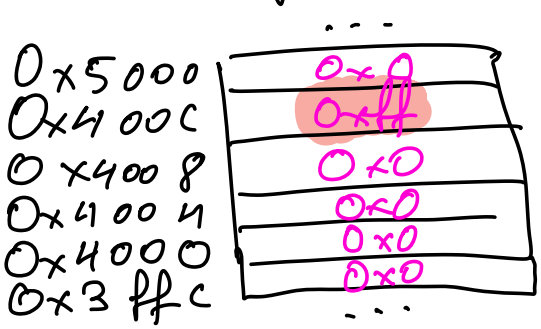
%eax = 0x4000, %ebx = 0x4, %ecx = 0x1, %edx = 0xff

mov -0xc(%eax, %ebx, 2), %ecx

mov %edx, 0x4(%ecx, %ebx)



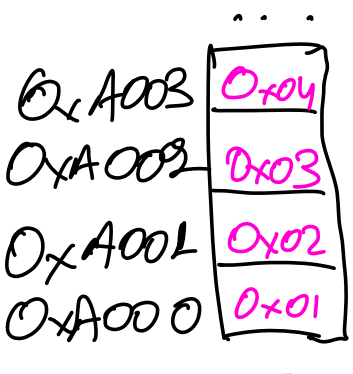
%ecx = 0x4004



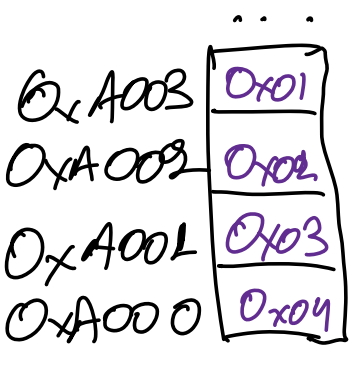
•) Endianness

0x01020304(H)

↳ 16909060(d)

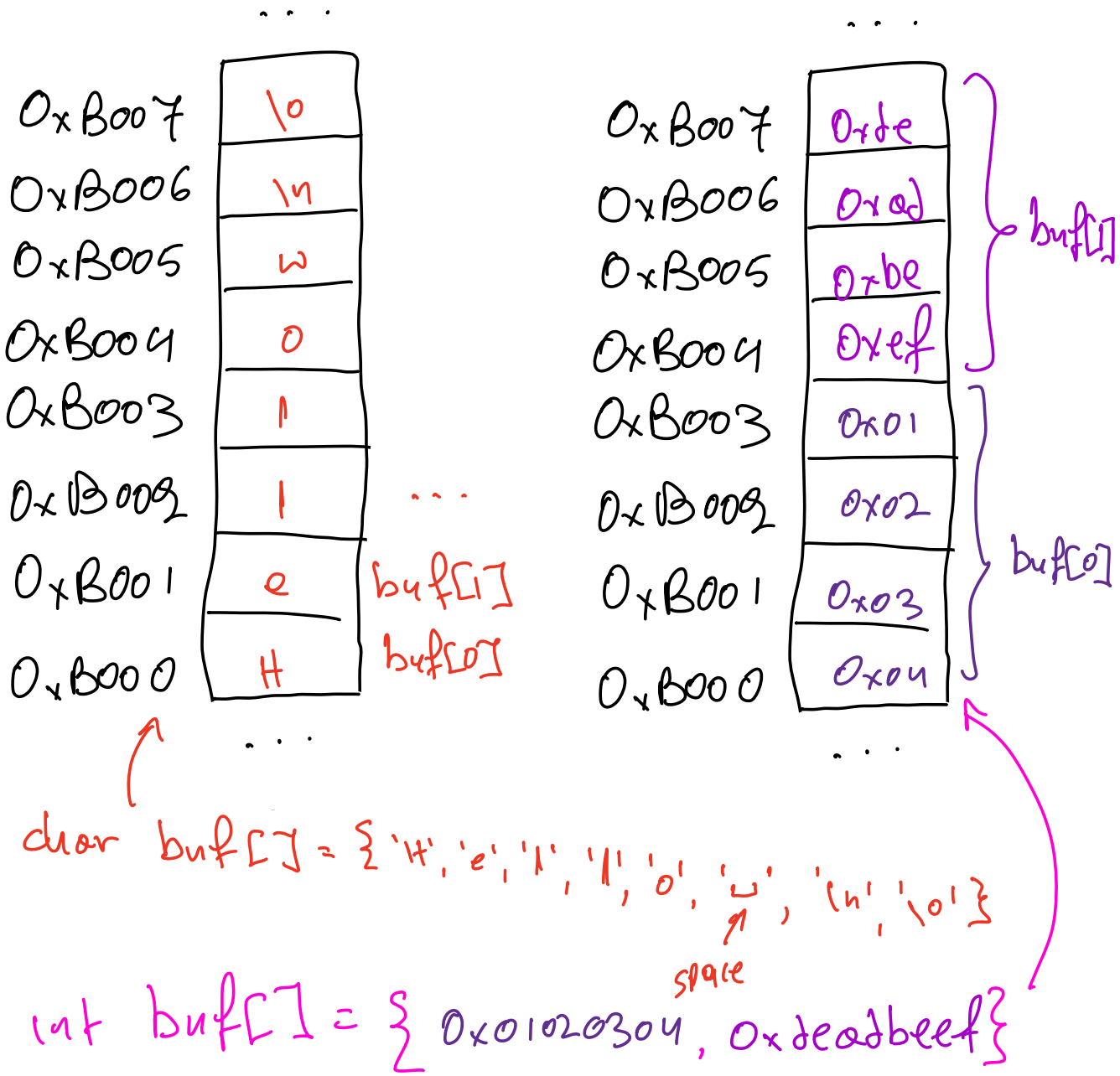


Big-endian



Little-endian

(x86)


0xB007
0xB006
0xB005
0xB004
0xB003
0xB002
0xB001
0xB000
\0
\n
w
o
l
l
e
H
buf[1]
buf[0]
0xde
0xad
0xbe
0xef
0x01
0x02
0x03
0x04
buf[1]
buf[0]
char buf[] = {'H', 'e', 'l', 'l', 'o', ' ', 'w', '\n', '\0'}
space
int buf[] = {0x01020304, 0xdeadbeef}