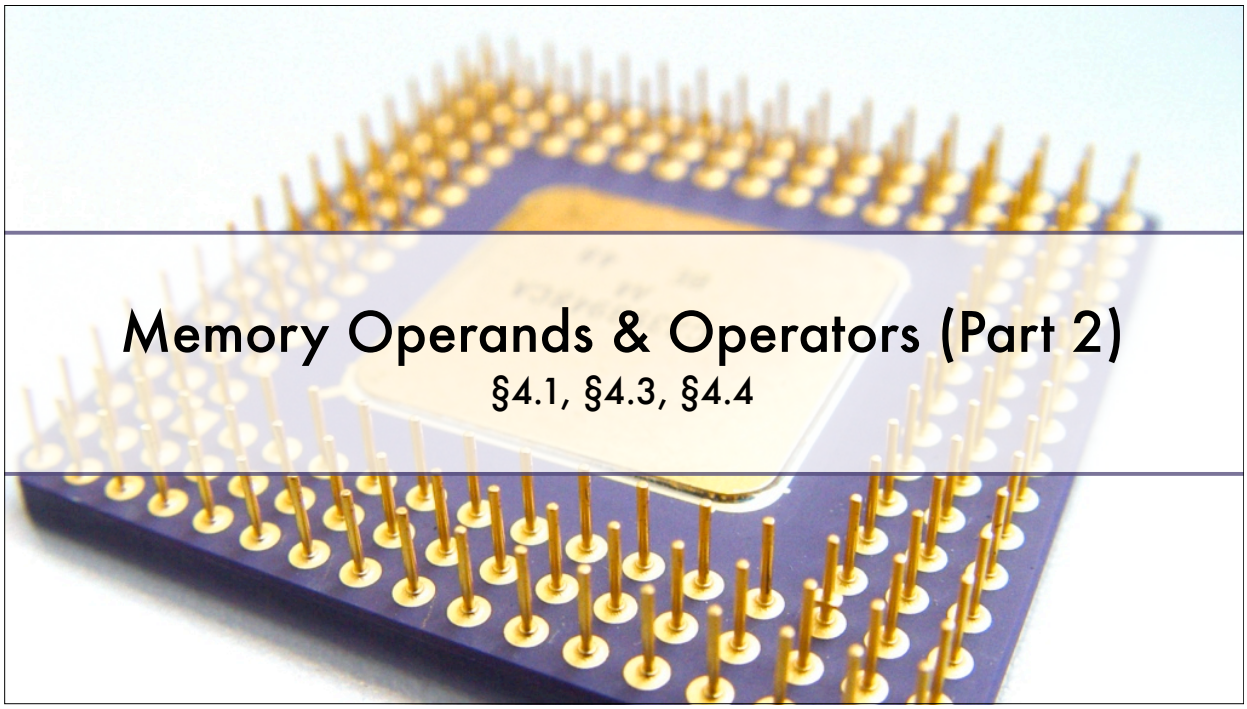
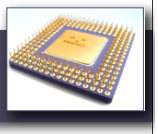


# Memory Operands & Operators (Part 2)

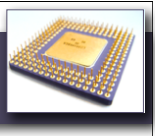
§4.1, §4.3, §4.4

## Homework



- Homework 3 was due in Canvas 11:00 a.m. today
- Turn in **Lab 4** no later than **Friday, 11:00 a.m.**
- For next class (Friday, October 10):
  - Read **Section 4.2.4** and **Sections 4.3.1–4.3.6**
    - Be able to describe what the NEG instruction does and what flags are affected
    - Be able to describe what the ALIGN directive does and why it's useful
    - Be prepared to verbally answer review questions 7–10 in §4.3.8 (p. 117)
  - Skim **Table 5-1** (6/e pp. 134–135, 7/e pp. 156–157)
    - Get an idea of what's provided by Kip Irvine's library (the "Irvine32" library)
    - Details of each procedure are in 6/e §5.3 (pp. 134–156), 7/e §5.4 (pp. 155–177)
- Homework 4 coming soon

# Review from Last Wednesday



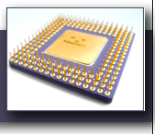
- Every memory operand has one or more parts of this general form:

[base + (index*scale) + displacement]

reg32 — reg32 (but not ESP) — 1, 2, 4, or 8 — 32-bit constant (data label or data label + constant)

- **Last Wednesday:**
  - LENGTHOF, SIZEOF operators
  - Direct Memory Operands — displacement only: data label
  - Direct-Offset Operands — displacement only: data label + constant
  - Indexed Operands — displacement + index
  - Scaled Indexed Operands — displacement + index*scale
- More memory operands later...

# Whiteboard Notes



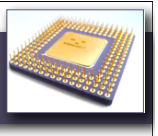
- Every memory operand has one or more parts of this general form:

[base + (index*scale) + displacement]

reg32 — reg32 (but not ESP) — 1, 2, 4, or 8 — 32-bit constant (data label or data label + constant)

- **Whiteboard Notes:**
  - OFFSET Operator
  - Indirect Operands — base only
    - PTR Operator
- More memory operands later...

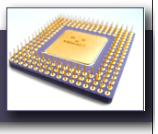
# Example 1: strlen



```
INCLUDE Irvine32.inc

.data
hi    BYTE "Hello", 0
bye   BYTE "See you", 0
crlf_ BYTE 0Dh, 0Ah, 0
empty BYTE 0

.code
strlen PROC
; Returns the length of a null-terminated string
; Receives: EDX -- Pointer to string
; Returns:  EAX -- Length of string

      TODO: Fill this in

strlen ENDP
```

```
main PROC
    mov edx, OFFSET hi
    call strlen     ; Returns 5 in EAX
    call WriteDec   ; Prints 5

    mov edx, OFFSET hi
    call strlen     ; Returns 7 in EAX
    call WriteDec   ; Prints 7

    mov edx, OFFSET crlf_
    call strlen     ; Returns 2 in EAX
    call WriteDec   ; Prints 2

    mov edx, OFFSET empty
    call strlen     ; Returns 0 in EAX
    call WriteDec   ; Prints 0

    exit
main ENDP
end main
```

# Example 2: min



```
INCLUDE Irvine32.inc

.data
ordered SDWORD -3, -2, -1, 0, 1, 2, 3
reverse SDWORD 3, 1, -1, -5
random  SDWORD 4, 8, 2, 7
single  SDWORD 3

.code
min PROC
; Returns the minimum value in an SDWORD array
; Receives: ESI -- Pointer to array
;           ECX -- Length of array (must be ≥ 1)
; Returns:  EAX -- Minimum value

      TODO: Fill this in

min ENDP
```

```
main PROC
    mov esi, OFFSET ordered
    mov ecx, LENGTHOF ordered
    call min        ; Returns –3 in EAX
    call WriteInt   ; Displays –3

    mov esi, OFFSET reverse
    mov ecx, LENGTHOF reverse
    call min        ; Returns –5 in EAX
    call WriteInt   ; Displays –5

    mov esi, OFFSET random
    mov ecx, LENGTHOF random
    call min        ; Returns 2 in EAX
    call WriteInt   ; Displays +2

    mov esi, OFFSET single
    mov ecx, LENGTHOF single
    call min        ; Returns 3 in EAX
    call WriteInt   ; Displays +3

    exit
main ENDP
end main
```