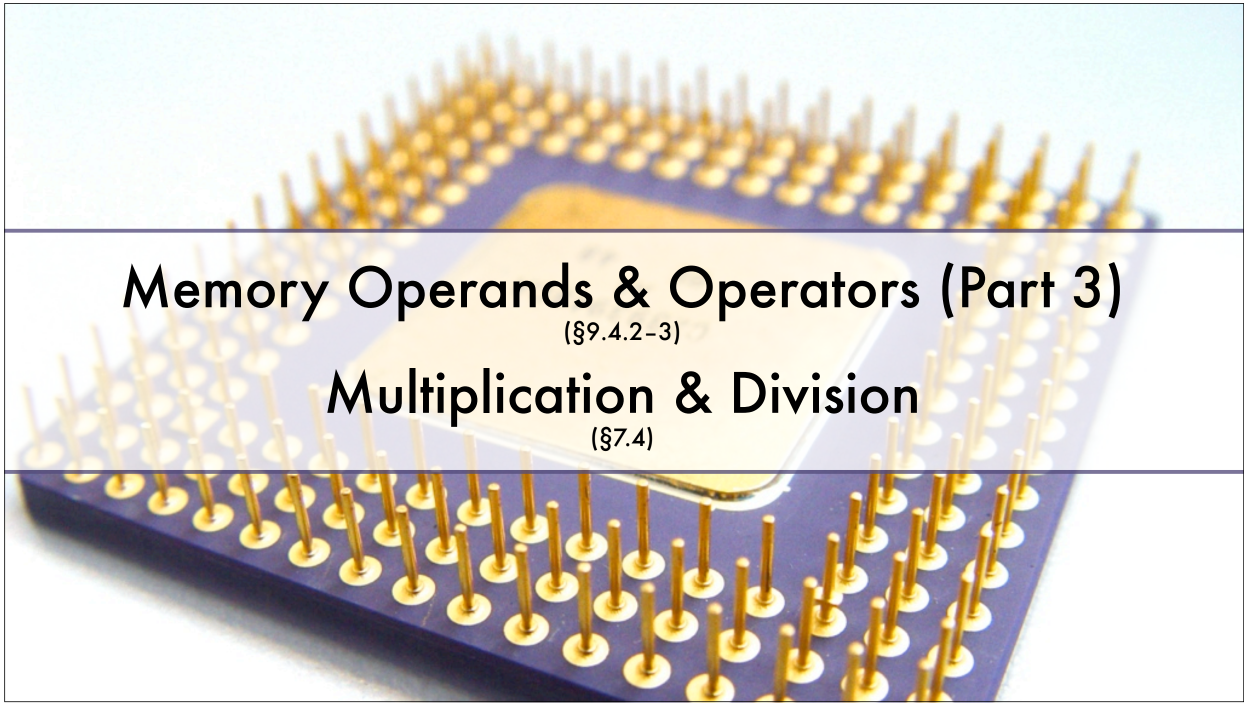


Memory Operands & Operators (Part 3)

(§9.4.2–3)

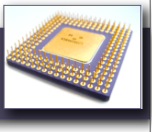
Multiplication & Division

(§7.4)

Homework

- For next class (Monday, October 13):
  - Read **Section 7.4,** skipping §7.4.3
  - Note 16- and 8-bit forms of the instructions we covered today
  - Be prepared to verbally answer
    - Questions 7, 8, 9, 11, 14, 15 in §7.4.7 (pp. 255–256)
    - Could you use MOVSX instead of CBW? CWD? CDQ?
    - *(Bonus)* When Visual C++ compiles a C++ program to assembly language, the assembly code it generates only uses IMUL and IDIV, even for unsigned arithmetic. Why does this work, since they're supposed to be *signed* arithmetic instructions?
- **Homework 4** will be posted this weekend

Memory Operands



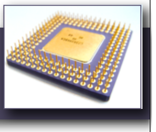
- Every memory operand has one or more parts of this general form:

$$\underset{\substack{\uparrow \\ \text{reg32}}}{[\text{base}} + (\underset{\substack{\uparrow \\ \text{reg32} \\ \text{but not ESP}}}{\text{index}} * \underset{\substack{\uparrow \\ \text{1, 2, 4, or 8}}}{\text{scale}}) + \underset{\substack{\uparrow \\ \text{32-bit constant} \\ \text{(data label or data label + constant)}}}{\text{displacement}]}$$

LENGTHOF, SIZEOF operators		
Direct Memory Operands	[array]	displacement only: data label
Direct-Offset Operands	[array + 2]	displacement only: data label + constant
Indexed Operands	[array + ecx]	displacement + index
Scaled Indexed Operands	[array + 2*ecx]	displacement + scale*index
Indirect Operands	[esi]	base
Base-Index	[esi + ecx] [esi + 2*ecx]	base + index base + index
Base-Index-Displacement	[esi + 2*ecx + 2]	base + scale*index + displacement

} **New Today**

Example 1: sumFirstLast



```
INCLUDE Irvine32.inc

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

.code
sumFirstLast PROC
; Returns the sum of the first and last elements
;    in an SDWORD array

; Receives: ESI -- Starting address of array
;           ECX -- # of elements in the array

; Returns:  EAX -- Sum of first and last elements

    TODO: Fill this in

sumFirstLast ENDP
```

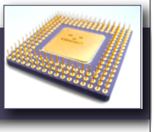
```
main PROC
    mov esi, OFFSET ordered
    mov ecx, LENGTHOF ordered
    call sumFirstLast
    call WriteInt   ; Prints –3

    mov esi, OFFSET random
    mov ecx, LENGTHOF random
    call sumFirstLast
    call WriteInt   ; Prints +6

    mov esi, OFFSET single
    mov ecx, LENGTHOF single
    call sumFirstLast
    call WriteInt   ; Prints +6 (= 3 + 3)

    exit
main ENDP
end main
```

Example 2: avgFirstLast



```
INCLUDE Irvine32.inc

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

.code
avgFirstLast PROC
; Returns the average of the first and last elements
;    in an SDWORD array
; Receives: ESI -- Starting address of array
;           ECX -- # of elements in the array
; Returns:  EAX -- Sum of first and last elements

    TODO: Fill this in

avgFirstLast ENDP
```

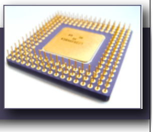
```
main PROC
    mov esi, OFFSET ordered
    mov ecx, LENGTHOF ordered
    call avgFirstLast
    call WriteInt   ; Prints –3

    mov esi, OFFSET random
    mov ecx, LENGTHOF random
    call avgFirstLast
    call WriteInt   ; Prints +6

    mov esi, OFFSET single
    mov ecx, LENGTHOF single
    call avgFirstLast
    call WriteInt   ; Prints +3

    exit
main ENDP
end main
```

Topics Covered in Notes:



- 32-bit forms of:
  - MUL instruction
  - IMUL instruction
  - DIV instruction
  - IDIV instruction
- CDQ instruction