Efficiency

CSC 236

- Miscellaneous Topics
- More File I/O
- Efficiency

For loop testing

- `for i=10 to 1 do { ... }`

```
        mov   ax,10      ;i = 10
again: …
        …
        sub   ax,1       ;i = i - 1
        ja    again      ;loop if i > 0
```

Did not test for overflow.

Is that OK?

Home key

- Pressing the “home” key
  - Returns “G”
  - In the key program
- Home key
  - *Extended ASCII* character
  - 2-byte character ⇒ 00_{16} 47_{16}
  - Generates two interrupts
  - System call reads second of the bytes

Write a string

- DOS end of string (EOS)
  - $ — dollar
  - Similar to C — '\0'
- Example

```
.data
msg    db   'ABC',13,10,'$'
```

41	42	43	0D	0A	24

Write a string

- DOS end of string (EOS)
  - $ — dollar
  - Similar to C — '\0'
- Example

```
.data
msg   db   'ABC',13,10,'$'
```

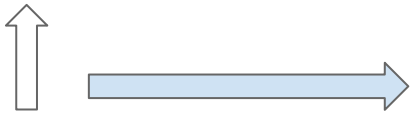
- System call semantics
  - ah = 9
  - dx = start of string

```
mov   ah, 9           ; write code
mov   dx, offset msg  ; pt to msg
int   21h             ; call dos
```

Write a string

- DOS end of string (EOS)
  - $ — dollar
  - Similar to C — '\0'
- Example

```
.data
msg   db   'ABC',13,10,'$'
```



offset msg

Prints from dx (offset msg) up to EOS ($)
does not print $.

- Int semantics
  - ah = 9
  - dx = start of string

```
mov   ah, 9           ; write code
mov   dx, offset msg  ; pt to msg
int   21h             ; call dos
```

Write a string

- DOS end of string (EOS)
  - $ — dollar
  - Similar to C — ‘\0’
- Example

```
.data
msg    db   'ABC',13,10,'$'
```

- What if you forget the ‘$’?

41	42	43	0D	0A	24

Write a string

- DOS end of string (EOS)
  - $ — dollar
  - Similar to C — ‘\0’
- Example

```
.data
msg    db   'ABC',13,10,'$'
```

- What if you forget the ‘$’?
- Writes what is in memory until
  - Encounters a byte = 24h
  - Ctl-Alt-Delete
  - Lose power

41	42	43	0D	0A	24

ASCII text file

In editor, you type

`1 2 <space> A <enter>`

`a <space> 9 <save>`

In DOS:

31	32	20	41	0D	0A	61	20	39	0D	0A	1A

- **True DOS — our grading system**

31	32	20	41	0D	0A	61	20	39	0D	0A	1A

- **DOS editor — *edit***

31	32	20	41	0D	0A	61	20	39	0D	0A

- **Unix editor — *vim***

31	32	20	41	0A	61	20	39	0A

- **Windows editor — wordpad**

31	32	20	41	0D	0A	61	20	39

Why do you care?

- The testing program
  - testfile file
  - Outputs hex for every character
  - Including control characters
- Programs to help
  - Compare two files: compfile file1 file2
  - Detect EOL and EOF characters: filerw < infile > outfile

Efficiency

- Definition
  - Completion of task
  - Using minimum resources
- Resources
  - Size
  - Time

Efficiency

- Definition
  - Completion of task
  - Using minimum resources
- Resources
  - Size
  - Time
  - Programmers time

Three null programs

C++	Java	Assembler

`void main( )`
`{ }`
	`class testsize`
	`{`
	`  public static`
	`  void main ( )`
	`  { }`
	`}`
		`x:  mov ax,4c00h`
		`  int 21h`
		`  end x`

Results

	Executable size (bytes)
C++ Unix	
C++ Windows	
Java on Unix	
ASM on Windows	

Results

	Executable size (bytes)
C++ Unix	
C++ Windows	
Java on Unix	
ASM on Windows	5

Results

	Executable size (bytes)
C++ Unix	
C++ Windows	
Java on Unix	236
ASM on Windows	5

Results

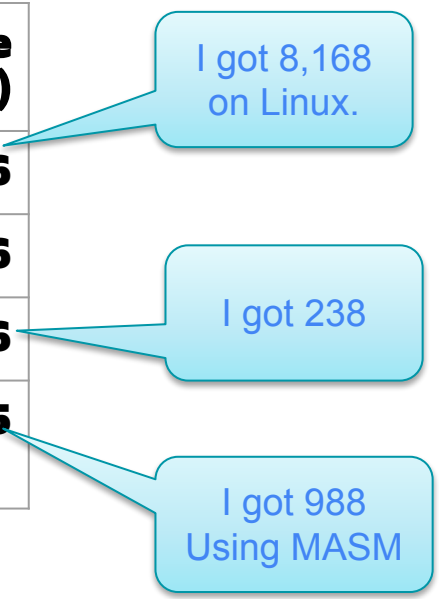
	Executable size (bytes)
C++ Unix	
C++ Windows	7,536
Java on Unix	236
ASM on Windows	5

Results

	Executable size (bytes)
C++ Unix	24,296
C++ Windows	7,536
Java on Unix	236
ASM on Windows	5

Results

	Executable size (bytes)
C++ Unix	24,296
C++ Windows	7,536
Java on Unix	236
ASM on Windows	5



Time

	key.c	key.asm
Instructions executed	45	20

Efficiency targets

- Programming assignments
  - Have efficiency targets
  - Rewards greater effort and understanding
- For example, key:

Instructions written		Instructions executed
00-20	20 pts	Not graded
21-22	15 pts	
23-24	10 pts	
25-26	05 pts	
27+	00 pts	

Efficiency

- No “Big Book of Efficiency”
  - A bit of an art
- Tips
  - Working code is #1 priority
  - Read chapter 23 in the notes
  - Think about efficiency in the design stage
  - Ask staff

Basic hints

#1 — Avoid unnecessary jumps

```
        cmp  al, '.'       ; is char a period
        je   exit          ; yes, exit program
        jmp  print         ; no,  goto print
;-------------------------------------------------
;   print the character
;-------------------------------------------------
print:
        mov  dl, al        ; put char in dl
        mov  ah, 2         ; set dos code
        int  21h           ; write
```

#1 — Avoid unnecessary jumps

```
        cmp  al, '.'        ; is char a period
        je   exit           ; yes, exit program
        jmp  print          ; no,  goto print
;-----------------------------------------------
;   print the character
;-----------------------------------------------
print:
        mov  dl, al         ; put char in dl
        mov  ah, 2          ; set dos code
        int  21h            ; write
```

#2 — Avoid unnecessary jumps II

```
        cmp  al, ' '      ; is char a space
        je   print        ; yes, print
        jmp  nexttest     ; no,  test next option
;-------------------------------------------------
;   print the character
;-------------------------------------------------
print:
        mov  dl, al       ; put char in dl
        mov  ah, 2        ; set dos code
        int  21h          ; write
```

#2 — Avoid unnecessary jumps II

```
        cmp  al, ' '       ; is char a space
        je   print         ; yes, print
        jmp  nexttest      ; no,  test next option
;-------------------------------------------------
;   print the character
;-------------------------------------------------
print:
        mov  dl, al        ; put char in dl
        mov  ah, 2         ; set dos code
        int  21h           ; write
```

#2 — Avoid unnecessary jumps II

```
        cmp  al, ' '       ; is char a space
        je   print         ; yes, print
        jmp  nexttest      ; no,  test next option
```

jne

```
;-------------------------------------------------
;   print the character
;-------------------------------------------------
print:
        mov  dl, al        ; put char in dl
        mov  ah, 2         ; set dos code
        int  21h           ; write
```

#3 — Avoid jumping around

Spaghetti code

#3 — Avoid jumping around

Spaghetti Code

```
     cmp  [val],0
     jl   err
ok:  ...
     ...
     ...


err: mov  [flag],1
     jmp  ok
```

Better

```
     cmp  [val],0
     jge   ok
     mov  [flag],1
ok:  ...
     ...
     ...
```

#4 — Initialize when needed

```
mov ax, 0    ; clear work register
mov ah, 8    ; code to read char
int 21h      ; call dos to read
```

#4 — Initialize when needed

```
mov ax, 0    ; clear work register
mov ah, 8    ; code to read char
int 21h      ; call dos to read
```

- Serves no purpose

Subtraction problem

Real subtraction

```
  7D550
- 5550
  78000
```

Two's complement subtraction

```
  7D550
  AAAF
+    1
  88000
```

Why are they different?

This isn't *fixed precision*

Subtraction problem

Real subtraction

```
  7D550
- 05550
  78000
```

Two’s complement subtraction

```
    1
  7D550
  FAAAF
+    1
  78000
```

Problem #1

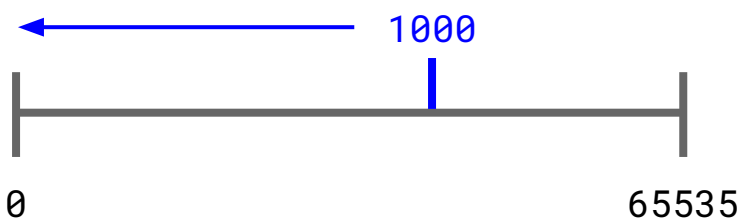
What is the range of values of var that jump to hit?

```
var     dw     ? ;unsigned word

cmp     [var],1000
jbe     hit
cmp     [var],100
jae     hit
jmp     miss
```

Problem #1

What is the range of values of var that jump to hit?

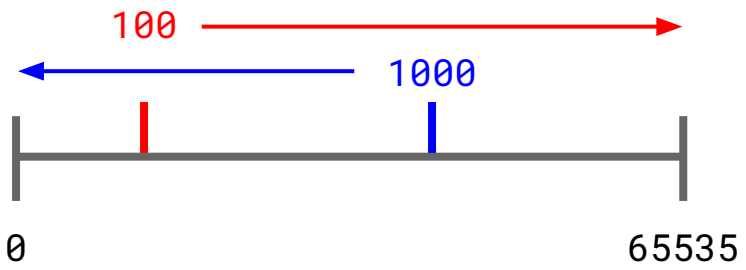
```
var     dw     ? ;unsigned word

cmp     [var],1000
jbe     hit
cmp     [var],100
jae     hit
jmp     miss
```



Problem #1

What is the range of values of var that jump to hit?

```
var     dw    ? ;unsigned word

cmp     [var],1000
jbe     hit
cmp     [var],100
jae     hit
jmp     miss
```



Problem #2

What is the value in al when code reaches fin?

```
        mov   al,125
repit:  add   al,1
        cmp   al,127
        jle   repit
fin:
```

```
125 = 7Dh
126 = 7Eh
126 ? 127
126 ≤ 127
loop
```

Problem #2

What is the value in al when code reaches fin?

```
        mov   al,125
repit:  add   al,1
        cmp   al,127
        jle   repit
fin:
```

127 = 7Fh
127 ? 127
127 ≤ 127
loop

Problem #2

What is the value in al when code reaches fin?

```
        mov   al,125
repit:  add   al,1
        cmp   al,127
        jle   repit
fin:
```

128 = 80h

128 ? 127

128 ≤ 127

80h ≤ 7Fh

Problem #2

What is the value in al when code reaches fin?

```
        mov   al,125
repit:  add   al,1
        cmp   al,127
        jle   repit
fin:
```

128 = 80h

128 ? 127

128 ≤ 127

80h ≤ 7Fh

signed test

negative

positive

Answer: Loops forever

Problem #3

What is the value in the al register when this code reaches fin?

```
        mov     al,253
repit:  inc     al
        jnc     repit
fin:
```

Problem #3

What is the value in the al register when this code reaches fin?

```
        mov     al,253
repit:  inc     al          mov does not set CF
        jnc     repit       inc does not set CF
fin:
```

Answer:

- **If CF = 1 upon entry**
  - **Executes once**
  - **al is 254**
- **If CF = 0 upon entry**
  - **Loops forever**