Το τελικό πρόγραμμα (i)

- Σκελετός:

```
xseg  segment  public 'code'
      assume   cs : xseg, ds : xseg, ss : xseg
      org      100h
main  proc     near
      call     near ptr program
      mov      ax, 4C00h
      int      21h
main  endp
```

... τελικός κώδικας που παράγεται ...

```
xseg  ends
      end      main
```

Το τελικό πρόγραμμα (ii)

- Βιβλιοθήκη χρόνου εκτέλεσης (run-time library)

```
extrn function : proc
```

- Σταθερές συμβολοσειρές και κινητής υποδιαστολής

```
@str1   db 'this is'
        db 10
        db 'an example'
        db 0

@real1  dt 1e-10
@real2  dt 2.0
```

Βοηθητικές ρουτίνες (i)

- getAR(a) (φόρτωση διεύθυνσης ΕΔ)

```
mov  si, word ptr [bp + 4]
mov  si, word ptr [si + 4]
...
mov  si, word ptr [si + 4]
```

} $(n_{\text{cur}} - n_{\text{a}} - 1$ φορές)

Βοηθητικές ρουτίνες (i)

- getAR(a) (φόρτωση διεύθυνσης ΕΔ)

```
mov  si, word ptr [bp + 4]
mov  si, word ptr [si + 4]
...
mov  si, word ptr [si + 4]
```

$(n_{\text{cur}} - n_{\text{a}} - 1$ φορές$)$

- updateAL() (ενημέρωση συνδέσμων προσπέλασης)

```
(α)   push  bp                              αν n_p < n_x
(β)   push  word ptr [bp + 4]               αν n_p = n_x
(γ)   mov   si, word ptr [bp + 4]           αν n_p > n_x
      mov   si, word ptr [si + 4]
      ...
      mov   si, word ptr [si + 4]
      push  word ptr [si + 4]
```

$(n_p - n_x - 1$ φορές$)$

Βοηθητικές ρουτίνες (ii)

- load(R, a) (φόρτωση τελουμένου)

	Είδος του a	Κώδικας που παράγεται
(α1)	αριθμητική σταθερά	`mov R, a`
(α2)	λογική σταθερά `true`	`mov R, 1`
(α3)	λογική σταθερά `false`	`mov R, 0`
(α4)	σταθερά χαρακτήρα	`mov R, ASCII(a)`
(α6)	σταθερά `nil`	`mov R, 0`
(β1)	τοπική οντότητα: μεταβλητή, παράμετρος κατ' αξία, ή προσωρινή μεταβλητή	`mov R,` *size* `ptr [bp +` *offset*`]`
(β2)	τοπική οντότητα: παράμετρος κατ' αναφορά	`mov si, word ptr [bp +` *offset*`]` `mov R,` *size* `ptr [si]`

Βοηθητικές ρουτίνες (iii)

- load(R, a) (φόρτωση τελουμένου)

	Είδος του a	Κώδικας που παράγεται
(γ1)	μη τοπική οντότητα: μεταβλητή, παράμετρος κατ' αξία, ή προσωρινή μεταβλητή	getAR(a) mov R, *size* ptr [si + *offset*]
(γ2)	μη τοπική οντότητα: παράμετρος κατ' αναφορά	getAR(a) mov si, word ptr [si + *offset*] mov R, *size* ptr [si]
(δ)	$[x]$	load(di, x) mov R, *size* ptr [di]
(ε)	$\{x\}$	loadAddr(R, x)

Βοηθητικές ρουτίνες (iv)

- loadAddr(R, a) (φόρτωση διεύθυνσης τελουμένου)

	Είδος του a	Κώδικας που παράγεται
(α5)	σταθερή συμβολοσειρά	`lea R, byte ptr a`
(β1)	τοπική οντότητα: παράμετρος κατ' αξία, ή προσωρινή μεταβλητή	`lea R,` *size* `ptr [bp +` *offset*`]`
(β2)	τοπική οντότητα: παράμετρος κατ' αναφορά	`mov R, word ptr [bp +` *offset*`]`
(γ1)	μη τοπική οντότητα: παράμετρος κατ' αξία, ή προσωρινή μεταβλητή	getAR(a) `lea R,` *size* `ptr [si +` *offset*`]`
(γ2)	μη τοπική οντότητα: παράμετρος κατ' αναφορά	getAR(a) `mov R, word ptr [si +` *offset*`]`
(δ)	$[x]$	load(R, x)

Βοηθητικές ρουτίνες (v)

- Παρόμοια υλοποίηση για τις:
  - loadReal(a)
  - store(R, a)
  - storeReal(a)

Βοηθητικές ρουτίνες (v)

- Παρόμοια υλοποίηση για τις:
  - loadReal(a)
  - store(R, a)
  - storeReal(a)
- Ρουτίνες για ετικέτες τελικού κώδικα:
  - name(p) _p_*num*
  - endof(p) @p_*num*
  - label(n) @n
  - label(ℓ) @p_*num*_ℓ

Παραγωγή κώδικα (i)

- Τετράδα $:=, \mathrm{x}, -, \mathrm{z}$

  load(R, x) loadReal(x)
  store(R, z) storeReal(z)

Παραγωγή κώδικα (i)

- Τετράδα $:=, \mathrm{x}, -, \mathrm{z}$

  load(R, x) loadReal(x)
  store(R, z) storeReal(z)

- Τετράδα array, x, y, z

  load(ax, y)
  `mov` cx, *size*
  `imul` cx
  loadAddr(cx, x)
  `add` ax, cx
  store(ax, z)

Παραγωγή κώδικα (ii)

- Τετράδες $+, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $-, \mathrm{x}, \mathrm{y}, \mathrm{z}$

$\mathrm{load}(\mathtt{ax}, \mathrm{x})$	$\mathrm{loadReal}(\mathrm{x})$
$\mathrm{load}(\mathtt{dx}, \mathrm{y})$	$\mathrm{loadReal}(\mathrm{y})$
instr $\mathtt{ax}, \mathtt{dx}$	finstr $\mathtt{ST}(1), \mathtt{ST}(0)$
$\mathrm{store}(\mathtt{ax}, \mathrm{z})$	$\mathrm{storeReal}(\mathrm{z})$
$\mathit{instr} = \mathtt{add}$ ή $\mathtt{sub}$	$\mathit{finstr} = \mathtt{faddp}$ κ.λπ.

Παραγωγή κώδικα (ii)

- Τετράδες $+, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $-, \mathrm{x}, \mathrm{y}, \mathrm{z}$

load(ax, x)
load(dx, y)
instr ax, dx
store(ax, z)

loadReal(x)
loadReal(y)
finstr ST(1), ST(0)
storeReal(z)

instr = add ή sub

finstr = faddp κ.λπ.

- Τετράδες $*, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $/, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $\%, \mathrm{x}, \mathrm{y}, \mathrm{z}$

load(ax, x)
load(cx, y)
imul cx
store(ax, z)

load(ax, x)
cwd
load(cx, y)
idiv cx
store(ax, z)

load(ax, x)
cwd
load(cx, y)
idiv cx
store(dx, z)

Παραγωγή κώδικα (iii)

- Τετράδες $=, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $<>, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $<, \mathrm{x}, \mathrm{y}, \mathrm{z}$
  $>, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $<=, \mathrm{x}, \mathrm{y}, \mathrm{z}$ $>=, \mathrm{x}, \mathrm{y}, \mathrm{z}$

load(ax, x)
load(dx, y)
cmp ax, dx
instr label(z)

instr = je, jne, κ.λπ.

loadReal(x)
loadReal(y)
fcompp
fstsw ax
test ax, *value*
instr label(z)

value και *instr* από Πίνακα 9.1 σελ. 249

Παραγωγή κώδικα (iv)

- Τετράδα ifb, x, −, z

```
load(al, x)
or   al, al
jnz  label(z)
```

Παραγωγή κώδικα (iv)

- Τετράδα ifb, x, −, z

```
load(al, x)
or   al, al
jnz  label(z)
```

- Τετράδα jump, −, −, z

```
jmp  label(z)
```

Παραγωγή κώδικα (iv)

- Τετράδα ifb, x, −, z

  load(al, x)
  or al, al
  jnz label(z)

- Τετράδα jump, −, −, z

  jmp label(z)

- Τετράδα jumpl, −, −, z

  jmp label(z)

Παραγωγή κώδικα (iv)

- Τετράδα ifb, x, −, z

  load(al, x)
  or al, al
  jnz label(z)

- Τετράδα jump, −, −, z

  jmp label(z)

- Τετράδα jumpl, −, −, z

  jmp label(z)

- Τετράδα label, −, −, z

  label(z) :

Παραγωγή κώδικα (v)

- Τετράδα unit, x, −, −

```
name(x)  proc  near
         push  bp
         mov   bp, sp
         sub   sp, size
```

Παραγωγή κώδικα (v)

- Τετράδα unit, x, −, −

```
name(x)  proc  near
         push  bp
         mov   bp, sp
         sub   sp, size
```

- Τετράδα endu, x, −, −

```
endof(x) :  mov   sp, bp
            pop   bp
            ret
name(x)     endp
```

Παραγωγή κώδικα (vi)

- Τετράδα call, −, −, z

```
sub   sp, 2                    αν z είναι διαδικασία
updateAL()
call  near ptr name(z)
add   sp, size + 4
```

Παραγωγή κώδικα (vi)

- Τετράδα call, −, −, z

  sub sp, 2
  updateAL()
  call near ptr name(z)
  add sp, *size* + 4

  αν z είναι διαδικασία

- Τετράδα ret, −, −, −

  jmp endof(*current*)

Παραγωγή κώδικα (vii)

- Τετράδα par, x, y, −
  - αν y = V και x είναι 16 bit

    load(ax, x)

    push ax
  - αν y = V και x είναι 8 bit

    load(al, x)

    sub sp, 1

    mov si, sp

    mov byte ptr [si], al

Παραγωγή κώδικα (vii)

- Τετράδα par, x, y, – (συνέχεια)
  - αν y = V και x είναι 80 bit

```
loadReal(x)
sub   sp, 10
mov   si, sp
fstp  tbyte ptr [si]
```

  - αν y = R ή RET

```
loadAddr(si, x)
push  si
```