Declararea datelor, definire constante, operanzi

Modificat: 22-Oct-23

De citit:
Capitolele 4.1 - 4.4, 4.6 - 4.8, 6.1 - 6.3

Cuprins

- Sintaxa instrucțiunilor
- Reguli sintactice
- Semnificatia entitatilor unei linii de program
- Moduri de adresare
- Declararea variabilelor
- Pseudo-operatori
- Macro-uri

Declarații

1. Directive
   - » Îi spun asamblorului ce să facă
   - » Ne-executabile
   - » Nu generează cod mașină
2. Instrucțiuni
   - » Îi spun CPU-ului ce să facă
   - » Operațiune + operanzi
3. Macrouri
   - » Notație scurtă pentru un grup de declarații
   - » Substituții cu parametri

Directive

- Directive care descriu structura programului
  - text, data, rodata, bss
  - Import/export nume
- Reminder(curs 2): structura procesului
- Va urma(curs 6): structura binarului

Variabile globale

- Trei zone de memorie
  - .data: initializate
  - .bss: neinitializate (zero @load-time)
  - .rodata: initializate, read-only
- .data: int a = 10;
- .bss: int a;
- .rodata: const int a = 10;
- Variabile locale declarate '**`static`**' in C

Directiva GLOBAL

- Directiva GLOBAL marcheaza etichetele vizibile global
  - » etichetele pot fi accesate si din alte module ale programului
- Formatul este

```
global    label1, label2, . . .
```

- Aproape orice label poate fi declarat global
  - » Nume de proceduri
  - » Nume de variabile
  - » equated labels
  - * Intr-o constructie GLOBAL, nu este necesar sa mentionam tipul labelului

Directiva EXTERN

- Directiva EXTERN ii spune asamblorului ca anumite labeluri nu sunt definite in modulul curent
  - Asamblorul rezerva spatiu in fisierul obiect pentru a fi utilizat ulterior de linker
- Formatul este

  `extern  label1, label2, . . .`

  unde **`label1`** si **`label2`** sunt declarate global folosind directiva GLOBAL in alte module

Instrucțiuni

[eticheta:] mnemonic [operanzi] [;comentariu]

- eticheta – șir de litere si cifre, care începe cu o literă
- mnemonic – nume care simbolizează o instrucțiune
  - `nop`
  - `jz begin`
  - `add eax, 1`

  `begin:`
  - `mov [buffer + ebx], ax`
- operanzi – registru | locație memorie| imediat
  - 0 – 2 operanzi

Sintaxa instrucțiunilor

- registru := EAX|EBX|AX|BX|AL|BL|..|ESI|..|CS|DS..|GS
- imediat:= număr sau expresie evaluată de asamblor
  - Adresa unei variabile este un imediat
- locație:= [expresie care produce o adresă logică]
  - Între paranteze drepte [expresie]
  - [variabilă + reg_index + reg_bază + deplasament]
  - deplasament = imediat
- **`mov [ceva + esi + ebx*4 + 9], ax`**
  - Locație = adresa variabilei ceva
  - Locație += 9; ceva + 9 este cunoscut la momentul asamblării
  - Locație += (ESI + EBX << 2)
  - La adresa indicată de Locație se stochează 16 biți din AX

Sintaxa instrucțiunilor x86

* Instrucțiuni fara operand
  - » NOP
  - » MOVSB
* instrucțiuni cu un operand
  - » PUSH EAX
  - » ROR DX
* instrucțiuni cu doi operanzi
  - » MOV AX, BX
* linie cu eticheta instrucțiune si comentariu
  - » START: MOV AX,BX ;mută conținut AX in BX
* linie de comentariu
  - » ; aceastaeste o linie de comentariu
* linie cu eticheta
  - » ETICHETA:

Reguli sintactice

- o singură instrucțiune/directivă per linie
- o linie poate conține:
  - nici o entitate de instrucțiune (camp) – linie goală
  - Numai etichetă
  - Numai comentariu
  - eticheta, instrucțiune, directivă și comentariu
- Comentariu incepe cu ';' si se încheie la sfârșitul liniei
- o instrucțiune x86 poate conține maxim 2 operanzi:
  - op1 – destinația și primul termen al operației
  - op2 – sursa sau al doilea termen al operației

Reguli sintactice - Exemple

* NOP
  - » Instrucțiune fara operanzi
* MOVSB
  - » instrucțiune cu operanzi impliciti
* MUL CL
  - » instrucțiune cu primul operand implicit (AX := AL * CL)
* MOV AX, BX
  - » AX := BX adică AX – destinatia , BX sursa transferului
* INC SI
  - » SI := SI + 1
* ADD CX, DX
  - » CX := CX + DX
* ADD AX, BX, CX
  - » Instrucțiune incorecta, prea multi operanzi

Reguli sintactice

- “case insensitive”
- Separarea câmpurilor se face cu spații, TAB-uri
- Pentru lizibilitate: separare coloane cu TAB:

  eticheta: mnemonic operanzi ;comentariu
- Nume simbolice în loc de valori numerice
  - adrese de variabila =>nume_variabila;
  - adrese de instrucțiune =>eticheta;
  - valori de constante numerice=>nume_constanta

Reguli sintactice- simboluri

- Simboluri, identificatori, etichete:
  - secventa de litere, cifre si unele caractere speciale (ex: _, $, @), ?), care nu incepe cu o cifra
  - Lungimea simbolului este arbitrara, dar se considera primele 31 caractere
  - Exista simboluri rezervate, predefinite in limbaj (cuvinte cheie, mnemonice, directive, nume registre)
  - exemple:
    - L1 BletchRightHereRight_Here Item1 __Special
    - $1234 @Home $_1 Dollar$ WhereAmI? @1234
  - erori:
    - 1TooMany – incepe cu o cifra
    - Hello.There – contine punct
    - $ - $ sau ? nu poate sa apara singur
    - LABEL – cuvant rezervat.

Reguli sintactice - constante

- Constante:
  - intregi: 12, 21d, 123h, 0fffh, 1011b
  - reale (virgulă mobila): 1.5, 1.0e10, 23.1e-12
  - Șir de caractere: "text", 'Text', 'TEXT''TEXT'
  - Constante simbolice:
    - În asm: unu equ 1
    - În C: #define unu 1

Reguli sintactice - operanzi

- Operanzii trebuie să aibă aceeași lungime
  - exceptii: operatii de inmultire si impartire)
- celmult un operand de tip locatie de memorie
  - Cum facem operații între două locatii de memorie?
  - Operatiile între registre – viteză mare
- Instrucțiunile sunt structural atomice
  - Sunt independente între ele
  - **nu există forme de programare structurată**
  - Structurarea programului: la nivel logic, prin directive