Tema 1

- Deadline: Joi, 16.11.2017 23:55
- Data publicării: 02.11.2017, 23:55
- Data ultimei modificări: 12.11.2017, 10:57
- Responsabili:
  - Claudiu Ghioc [mailto:claudiu.ghioc@gmail.com]
  - Răzvan Cojocaru [mailto:razvan.cojocaru93@gmail.com]

Enunț

Să se implementeze un program în limbaj de asamblare care efectuează conversia unor numere într-o bază dată și afișează rezultatele. Numerele ce trebuie convertite sunt numere întregi, pozitive, pe 32 de biți, iar baza poate să fie între 2 și 16.

Implementare

Programul va folosi doi vectori, unul pentru numerele ce trebuie convertite și unul pentru bazele folosite la conversie. Se va converti numărul de pe poziția **i** din vectorul de numere în baza de pe poziția **i** din vectorul de baze. În final se vor afișa pe câte o linie numerele obținute, sub forma unor șiruri de caractere (0..9, a..f). În cazul în care baza nu este una validă se va afișa mesajul "Baza incorecta".

În momentul asamblării, programul va include un fișier ce conține un input format din trei variabile:

- dimensiunea vectorului de numere: variabila **nums**
- vectorul de numere ce trebuie convertite: variabila **nums_array**
- vectorul de baze în care se va face conversia: variabila **base_array**.

Acest fișier se va numi **input.inc** și trebuie să fie inclus în fișierul sursă ce conține implementarea temei, în secțiunea de date (imediat după section .data). Găsiți în arhiva cu resursele temei un fișier de la care puteți începe implementarea.

Exemplu de fișier de input:

```
nums dd 4
nums_array dd 612, 1330, 7, 12988
base_array dd 16, 1, 2, 14
```

Exemplu de fișier de output:

```
264
Baza incorecta
111
4a3a
```

Testare

Tema se poate testa pe platforma vmchecker sau local folosind checker-ul din arhiva cu resursele temei.

Arhiva conține o serie de fișiere de intrare în directorul **inputs** și fișiere ce conțin rezultatele așteptate pentru fiecare test, în directorul **outputs**. Verificarea acestor teste este făcută automat de către checker.

Fișierul **README** din arhivă conține instrucțiuni despre folosirea checker-ului.

Trimitere și notare

Temele vor trebui încărcate pe platforma vmchecker [https://vmchecker.cs.pub.ro/ui/#IOCLA] (în secțiunea IOCLA) și vor fi testate automat. Arhiva încărcată trebuie să fie o arhivă `.zip` care să conțină:

- fișierul sursă ce conține implementarea temei, denumit `tema1.asm`
- fișier `README` ce conține descrierea implementării

Punctajul final acordat pe o temă este compus din:

- punctajul obținut prin testarea automată de pe vmchecker - 80%
- coding style - 10%. Se va ține cont de:
  - claritatea codului
  - indentare coerentă
  - comentarii
  - nume sugestive pentru label-uri
- fișier README - 10%

Temele care nu trec de procesul de asamblare (*build*) nu vor fi luate în considerare.

Mașina virtuală folosită pentru testarea temelor de casă pe vmchecker este descrisă în secțiunea Mașini virtuale din pagina de resurse.

Precizări suplimentare

- Dacă folosiți SASM pe Windows, pentru a putea testa va trebui să puneți fișierul ce conține inputul, input.inc, în directorul include care se află în directorul în care este instalat SASM (de exemplu C:\Program Files (x86)\SASM\include).
- Metodele de conversie din baza 10 în altă bază sunt prezentate în laboratorul 1.
- Pentru afișarea unui caracter din rezultat (ex. "1234", "abcd") trebuie să folosiți macro-ul `PRINT_CHAR` din SASM (găsiți aici [https://dman95.github.io/SASM/english.html] mai multe detalii). Alte metode de afișare sunt depunctate.
- Pentru afișarea mesajului "Baza incorecta" puteți folosi macro-ul `PRINT_STRING` din SASM sau funcții precum `printf`, `puts`.
- Aici [http://ocw.cs.pub.ro/courses/iocla/bune-practici] puteţi găsi un cheatsheet, recomandări, o serie de bug-uri frecvente, etc.
- Împărțirea cu rest se poate efectua folosind instrucțiunea div, care funcționează conform tabelului de mai jos:

Deîmpărțit	Împărțitor	Cât	Rest
AX	Registru pe 8 biți	AL	AH
DX:AX	Registru pe 16 biți	AX	DX
EDX:EAX	Registru pe 32 de biți	EAX	EDX

Cu alte cuvinte, în funcție de dimensiunea deîmpărțitului și a împărțitorului, trebuie să plasăm numărul pe care dorim să îl împărțim în:

- AX - dacă dorim să împărțim la un număr pe 8 biți
- DX:AX (primii cei mai semnificativi 16 biți în DX, ultimii 16 în AX) - dacă dorim să împărțim un număr pe 32 biți la unul pe 16 biți

- EDX:EAX (primii 32 biți în EDX, ultimii în EAX) - dacă dorim să împărțim un număr pe 64 de biți la unul pe 32 de biți.

Registrul cu care dorim să împărțim este dat ca argument instrucțiunii div.

Exemple:

- **div BX** împarte valoarea din DX:AX la valoarea din BX și stochează câtul în AX și restul în DX.
- **div BH** împarte valoarea din AL:AH (AX) la valoarea din BH și stochează câtul în AL și restul în AH.

Resurse

Arhiva ce conține checker-ul, testele și fișierul de la care puteți începe implementarea este aici.

iocla/teme/tema-1.txt · Last modified: 2017/11/12 20:18 by constantin.ghioc