Tema - Arhitectura Sistemelor de Calcul

Seriile 13, 14, 15

Decembrie 2022

Cuprins

1 Detalii administrative **2**
- 1.1 Deadline . . . 2
- 1.2 Reamintirea punctajului pe tema . . . 2
- 1.3 Transmitere . . . 2
- 1.4 Ce se va transmite . . . 2
- 1.5 Cum se va face evaluarea . . . 2
- 1.6 Evaluarea automata a procedurilor . . . 2
- 1.7 Alte observatii . . . 3

2 Formularea temei **4**

3 Cerinte **6**
- 3.1 Cerinta 1 - 5p . . . 6
- 3.2 Cerinta 2 - 3p . . . 7
- 3.3 Cerinta 3 - 2p . . . 7

1 Detalii administrative

1.1 Deadline

Puteți trimite solutiile **cel tarziu pe 6 Ianuarie 2023, ora 23:55**.

1.2 Reamintirea punctajului pe tema

Tema valoreaza 25% din nota la acest laborator (conform Cursului 0x00), si este necesara obtinerea notei 5 pentru promovare.

1.3 Transmitere

Veti trimite solutiile in urmatoarele formulare, in functie de serie:

- seria 13: `https://forms.gle/oDLULw2ZC2AqAomAA`
- seria 14: `https://forms.gle/qTRw1LZNqPM5pRmz7`
- seria 15: `https://forms.gle/Xy3fpFPou7hbAKd3A`
- restantieri: `https://forms.gle/eeFUmi3xWCCzzoAz9`

1.4 Ce se va transmite

Se vor incarca in formular **doua surse** cu denumirea **grupa_nume_prenume_0.s** (pentru primele doua cerinte), respectiv **grupa_nume_prenume_1.s** (pentru a treia cerinta). Daca aveti mai multe nume / prenume, veti incarca surse de forma **172_GeorgescuXulescu_IonVasile_0.s**. Este **important** sa incarcati surse cu denumirea corecta, deoarece testarea va fi **automata**.

1.5 Cum se va face evaluarea

Exista doi pasi pentru obținerea notei:

- se verifica toate sursele sa nu fie cazuri de plagiat. In cazul in care se detecteaza plagiat, se face automat sesizare catre *Comisia de Etica a Universitatii din Bucuresti*;
- sursele care au trecut de verificarea anti-plagiat, vor fi testate automat.

Important! Studentii care au alte configurari fata de cele pe care lucram la laborator, trebuie sa precizeze acest lucru in formularul in care transmit tema, pentru a putea efectua evaluarea si pentru a nu primi 0 implicit.

1.6 Evaluarea automata a procedurilor

Atentie la modul in care implementati procedurile! In cadrul acestora, trebuie sa respectati **toate** conventiile de apel, si anume:

- argumentele trebuie incarcate pe stiva pentru apel, conform standardului `x86`, de la dreapta la stanga;
- registrii `%eax`, `%ecx` si `%edx` sunt singurii care **NU** trebuie restaurati in urma apelului, fiind si registrii de returnare; toti ceilalti registri **trebuie restaurati**!

- in cadrul de apel utilizam **registrul** `%ebp` conform conventiei `x86`, prezentata la laborator si in suportul 0x01 de laborator;
- in cadrul de apel **NU** se utilizeaza variabile din sectiunea `.data`! Toate variabilele au scop local, deci se vor afla **pe stiva**, si vor fi accesate prin intermediul lui `%ebp`!

Daca se incalca una sau mai multe dintre aceste conventii, se considera ca procedura nu a fost implementata corect.

De exemplu, daca avem nevoie in procedura de o variabila locala in care sa retinem o anumita valoare, varianta corecta este urmatoarea (consideram ca avem configuratia urmatoare a stivei - `%esp:%ebp:(%ebp vechi)(adresa de retur)(arg1)(arg2)...(argn)`):

```
subl $4, %esp           // alocam un spatiu pe stiva

// utilizam acum acest spatiu pe post de variabila locala
// el este -4(%ebp)
// %esp:(<spatiul declarat>)%ebp:(%ebp vechi)(adresa de retur)(arg1)  (arg2)...(argn)
//       -4(%ebp)                 0(%ebp)     4(%ebp)         8(%ebp) 12(%ebp) ...

movl $1, -4(%ebp)       // exemplu

addl $4, %esp           // nu uitam sa dezalocam spatiul
```

Este incorect sa avem o variabila in `.data`, de exemplu `x: .space 4`, si sa scriem in procedura:

```
movl $1, x
```

deoarece procedura **trebuie** sa fie generica, si **nu** trebuie sa depinda de ce variabile au fost declarate in programul din care este apelata.

Important! Scriptul de evaluare automata identifica daca sunt incalcate conventiile de apel de mai sus, asa ca este important sa le respectati.

1.7 Alte observatii

1. Nu va interzicem sa discutati idei intre voi, dar aveti grija, deoarece este o diferenta importanta intre a da o idee si a da codul direct.
2. Nu folositi convertoare automate din C/C++/alte limbaje in x86, le-am folosit si noi si recunoastem fara dificultate un cod care nu este scris de voi.

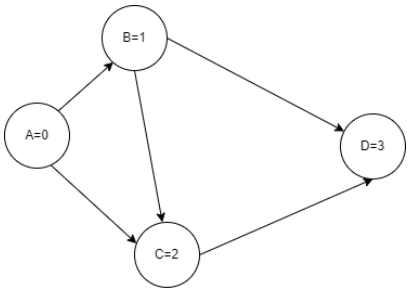
2 Formularea temei

Fie $\mathcal{G} = (V, E)$ un graf orientat, unde V este multimea varfurilor, E multimea muchiilor orientate, iar $AD(\mathcal{G})$ matricea de adiacenta asociata acestuia. Ne propunem sa determinam cate drumuri de lungime k (k indica numarul muchiilor) exista intre doua noduri i si j, $(i, j) \in V^2$, unde un drum este o secventa $i \to k_0 \to k_1 ... \to k_n \to j$, unde $(i, k_0) \in E \wedge (k_p, k_{p+1}) \in E, \forall p \in \{0, 1, ..., n-1\} \wedge (k_n, j) \in E$.

Pentru exemplificare, vom considera graful orientat de mai jos, definit prin:

$$\mathcal{G} = \big(V := \{A, B, C, D\}, E := \{(A, B); (A, C); (B, C); (B, D); (C, D)\}\big)$$

Reprezentarea vizuala a acestui graf este urmatoarea:



Matricea de adiacenta asociata este

$$AD(\mathcal{G}) := \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Daca vrem sa determinam cate drumuri de lungime $k = 1$ gasim intre doua varfuri, atunci gasim deja aceasta informatie in $AD(\mathcal{G})$.

Presupunem ca vrem sa determinam toate drumurile de lungime $k = 2$ din acest graf. Daca ne uitam strict la reprezentare, drumurile sunt urmatoarele:

- A - B - C
- A - B - D
- A - C - D
- B - C - D

Am gasit ca de la **A** la **C** exista un singur drum de lungime 2, de la **A** la **D** sunt doua drumuri de lungime 2, respectiv de la **B** la **D** exista un singur drum de lungime 2. Putem reprezenta aceasta informatie tot matriceal:

$$\mathcal{D}_2 = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Sa presupunem ca vrem sa determinam toate drumurile de lungime $k = 3$. Din nou, analizand reprezentarea grafului, gasim ca singura varianta este **A** - **B** - **C** - **D**. In reprezentare matriceala,

$$\mathcal{D}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Pentru orice alt $k \geq 4$, fiecare $\mathcal{D}_k = \mathbf{0}_4$ (matricea patratica 4×4 cu toate elementele egale cu 0), intrucat nu exista niciun drum de lungime 4 sau cu lungime mai mare decat 4.

Intrebarea este daca exista o relatie intre $AD(\mathcal{G})$ si $\mathcal{D}_2$, intre $AD(\mathcal{G})$ si $\mathcal{D}_3$ si, in general, intre $AD(\mathcal{G})$ si $\mathcal{D}_p$, pentru orice $p \in \mathbb{N}^*$.

Observam ca

$$\begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \mathcal{D}_2 := AD^2(\mathcal{G}) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Analog, observam ca

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \mathcal{D}_3 := AD^3(\mathcal{G}) = \mathcal{D}_2 \cdot AD(\mathcal{G}) = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Atunci, in caz general, intre nodurile $(i, j) \in V^2$ exista un drum de lungime k daca in $AD^k(\mathcal{G})$ pe pozitia (i, j) exista o valoare diferita de 0; in acest caz, valoarea ne indica exact numarul de drumuri existente.

3 Cerinte

In cadrul acestei teme aveti trei cerinte - o cerinta pentru 5p, una pentru 3p, respectiv o cerinta pentru alte 2p.

Important! NU dati inputul manual la fiecare retestare a programului! Sunt inputuri lungi, care va vor costa timp. Creati-va un fisier, de exemplu `input.txt`, in care scrieti inputul dorit, iar dupa ce aveti un executabil, de exemplu `cerinta1`, pe care in mod normal l-ati fi rulat cu `./cerinta1`, rulati comanda `./cerinta1 < input.txt`. Astfel, continutul din fisier va fi redirectat la **STDIN**, exact ca atunci cand ati fi introdus manual valorile. Folositi aceasta informatie si pentru a va testa mai multe inputuri, creandu-va fisiere `input0.txt`, `input1.txt` etc., si testandu-le cu `./cerinta1 < input0.txt`, `./cerinta1 < input1.txt` etc.

3.1 Cerinta 1 - 5p

Se citesc de la tastatura (**STDIN**) **listele de adiacenta** ale grafului orientat $\mathcal{G}$ si se cere sa se afiseze **matricea de adiacenta**. Astfel, se vor citi, cate **o valoare pe linie**:

- `1` reprezentand numarul cerintei; pentru cerintele 2 si 3, aceasta valoare va fi 2, respectiv 3;
- `N <= 100` numarul de noduri ale grafului;
- `N` linii pe care se vor afla valorile M_0, M_1, ..., M_{N-1}, reprezentand numarul de legaturi pentru fiecare nod in parte;
- M_0 linii pe care se vor afla vecinii nodului 0, apoi M_1 linii pe care se vor afla vecinii nodului 1, ..., apoi M_{N-1} linii pe care se vor afla vecinii nodului N-1.

De exemplu, pentru graful din Sectiunea 2, considerand nodurile A = 0, B = 1, C = 2, D = 3, inputul ar avea urmatoarea forma *(evident, fara comentariile din dreapta, acestea au fost puse pentru a va fi clar ce inseamna fiecare valoare)*:

```
1                   // numarul cerintei
4                   // nr. noduri
2                   // 0 are 2 legaturi (cu 1 si 2)
2                   // 1 are 2 legaturi (cu 2 si 3)
1                   // 2 are 1 legatura (cu 3)
0                   // 3 nu are nicio legatura
1                   //      legaturile
2                   //      nodului 0
2                   //      legaturile
3                   //      nodului 1
3                   //      legatura nodului 2
```

In urma primirii acestui input, se va afisa la **STDOUT** urmatorul output:

```
0 1 1 0
0 0 1 1
0 0 0 1
0 0 0 0
```

reprezentand matricea de adiacenta construita.

3.2 Cerinta 2 - 3p

Pentru aceasta cerinta, se vor citi de la **STDIN listele de adiacenta** exact ca la **Cerinta 1**, se va construi in spate **matricea de adiacenta** si se va calcula **numarul de drumuri de lungime k dintre doua noduri date**, unde k = lungimea, i = nodul sursa, respectiv j = nodul destinatie vor fi specificate in input.

Un exemplu de input este urmatorul *(ca la cerinta anterioara, inputul este format doar din valorile numerice din stanga, in dreapta sunt comentarii care sa va ajute sa intelegeti ce reprezinta fiecare valoare)*:

```
2                   // numarul cerintei
4                   // nr. noduri
2                   // 0 are 2 legaturi (cu 1 si 2)
2                   // 1 are 2 legaturi (cu 2 si 3)
1                   // 2 are 1 legatura (cu 3)
0                   // 3 nu are nicio legatura
1                   //      legaturile
2                   //      nodului 0
2                   //      legaturile
3                   //      nodului 1
3                   //      legatura nodului 2
2                   // lungimea drumului
0                   // nodul sursa
3                   // nodul destinatie
```

Outputul dat la **STDOUT** va fi, in acest caz, `2`. (exista 2 drumuri de la 0 la 3, respectiv de la A la D in formularea cerintei din sectiunea anterioara).

Pentru rezolvarea acestei cerinte **trebuie** sa implementati o procedura care sa respecte toate conventiile prezentate la laborator si in suportul de laborator, cu numele `matrix_mult`, care sa primeasca doua matrici ca input, patratice si de aceeasi dimensiune, si sa calculeze produsul, stocandu-l intr-un spatiu definit anterior in sectiunea `.data`. Signatura acestei proceduri va fi

```
matrix_mult(m1, m2, mres, n)
```

unde `m1` si `m2` sunt adresele matricelor in memorie, `mres` este adresa matricei in care se va completa rezultatul, iar `n` reprezinta dimensiunea acestora. Daca rezolvarea se face fara implementarea unei proceduri, sau daca sunt incalcate una sau mai multe dintre conventiile de implementare, **NU** veti obtine punctaj.

Important! Trebuie ca signatura sa fie respectata strict, **NU puteti alege un alt nume pentru procedura!** Veti avea, obligatoriu, o eticheta `matrix_mult` care va reprezenta adresa de inceput a procedurii care va rezolva inmultirea. In caz contrar, punctajul pe care il veti obtine la aceasta cerinta va fi 0.

3.3 Cerinta 3 - 2p

Pentru aceasta cerinta, refaceti Cerinta 2 (deci dati exact acelasi input, dar punand 3 pe prima linie - corespunzator numarului cerintei), dar de data aceasta alocati spatiul pentru matricea folosita **in mod dinamic**, folosind apelul de sistem `mmap2`. Aceasta operatie este echivalentul unui `malloc` din *libc*, iar pentru dealocare memoriei echivalentul utilizarii `free` din libc.

Codul pentru apelul de sistem `mmap2` este 192 si asteapta 6 parametri, care vor fi completati in registrii: eax (192), ebx (0), ecx (dimensiunea de alocat), edx (prot), esi (flags), edi (fd) si ebp (offset).

Pentru a obtine punctajul la aceasta cerinta, pe langa functionalitatea corecta a programului, trebuie sa scrieti un comentariu pe fiecare linie din apelul de sistem, in care sa explicati ce valoare ati pus in fiecare dintre acesti registri si **de ce**.

Gasiti detalii despre felul in care aceast syscall se apeleaza la urmatoarele resurse:

- Tabel de syscall-uri `https://marcin.juszkiewicz.com.pl/download/tables/syscalls.html`
- Pagina man pentru mmap2 `https://www.man7.org/linux/man-pages/man2/mmap2.2.html`