Miniprocesor

Sa se implementeze un interpretor de biti similar unui procesor. Acesta va avea capacitatea de a decodifica si executa instructiuni simple de adunare, scadere, inmultire si impartire.

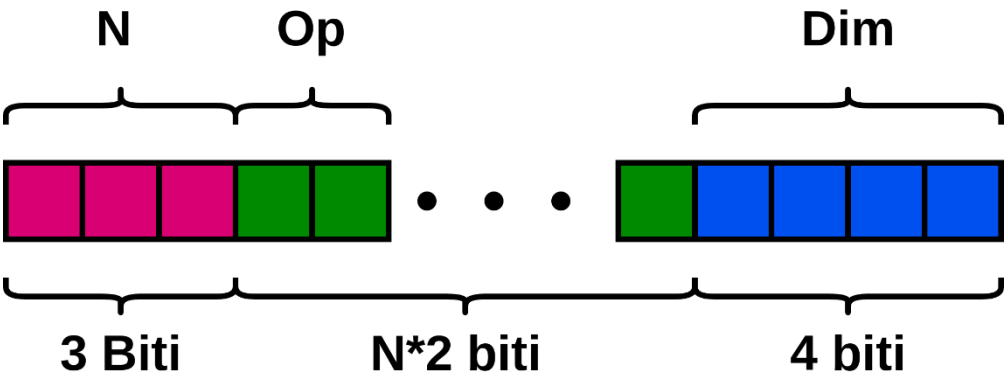
La nivelul cel mai de baza, informatia este stocata sub forma de biti. Pentru un procesor aceasta informatie se repartizeaza in 2 categorii: instructiuni si date. Practic, dandu-se un sir de biti, procesorul decodifica instructiunea, iar ulterior o executa.

In aceasta tema, vom implementa un procesor de baza care decodifica un sir de biti si ulterior il executa.

Task 1 - Decodificare instructiune

Dandu-se o instructiune in format binar, decodificati instructiunea.

O instructiune are urmatorul format:



Unde:

- **N** reprezinta numarul de instructiuni ce vor fi executate; acesta este reprezentat pe 3 biti si se obtine prin convertirea valorii celor mai semnificativi 3 biți din binar in decimal si adunarea cu 1. Astfel, pentru **000** vom avea de executat o instructiune, pentru **010** vom avea de executat 3 instrucituni. Numărul maxim de instrucțiuni de executat este 8.
- **Op** reprezintă codul unei instructiuni și se reprezintă pe 2 biți. **Op** poate fi **+**, **-**, * sau / conform tabelului de mai jos:

Cod	Operatie
00	+
01	-
10	*
11	/

In sirul de input, cei 3 biți care desemnează numărul de operații sunt urmați de număr de N*2 biți care desemnează operațiile ce urmează a fi executate.

- **Dim** reprezintă dimensiunea unui operand și se reprezintă pe 4 biti. **Dim** se calculează similar cu **N** prin transformarea celor mai puțini semnificativi 4 biți în valoarea zecimală și adunarea cu 1. Astfel, dimensiunea operanzilor poate lua valori din intervalul [1, 16].

In cadrul acestui exercițiu, veți citi de la intrarea standard un numar de tipul **unsigned int** ce contine instructiunea si o veti decodifica. Astfel, la iesirea standard veti afisa N, operatorii si dimensiunea operanzilor, toate separate printr-un spatiu.

Exemplu:
Input: 1675493376 → Output: 4 + - / * 16

Input: 2483421184 → Output: 5 * * + + + 13

Precizari:

- dimensiunea totala a unei instructiunie nu poate depasi 3 + 8*2 + 4 = 23 biti, asadar ar trebui sa incapa intr-un unsigned int.

- citirea de la tastatura se va face folosind functia scanf: **scanf("%u", &inst)**
- formatul de afisare este: **N op1 op2 .. opN Dim**
- rezolvarea acestui task se va afla in fisierul **task1.c**

Task 2 - Executare instructiune

În cadrul acestui exercițiu vom continua task-ul anterior prin:

1. **Citirea operanzilor**. Plecând de la programul anterior, adăugați o sectiune de cod care interpretează (N+1) operanzi de la intrarea standard. Pentru acest task, dimensiunea operanzilor (**Dim**) este un numar putere a lui 2 din intervalul [1, 16]. Adică valorile posibile sunt: 1, 2, 4, 8, 16. Operanzii vor fi cititi sub forma unor numere **unsigned short** (dimensiune 16) de la intrarea standard. Numărul de operanzi citiți de la tastatură se va descompune în mai multe de numere **unsigned short**, folosind formula: ((N+1)*Dim)/16, la care adaugam +1 in cazul in care rezultatul are vreun rest. Astfel, se vor citi de la tastatură ((N+1)*Dim)/16 numere și vor fi descompune în (N+1) operanzi.
Exemplu:
Pentru N = 3 si Dim = 4, folosim instrucțiunea `1410859008`, output-ul de la Task 1 va fi `3 * * * + 4`. Conform formulei de mai sus `((N+1)*Dim)/16`, vom citi un singur **unsigned short** de la tastatura (((3+1)*4)/16 = 1). Presupunem că vom citi 54999. Valoarea acestuia in binar este: **1101 0110 1101 0111**. Practic, primul operand va fi 13, al doilea 6, al treila 13, iar al patrulea 7.
Daca N = 4 si Dim = 8, folosim instrucțiunea `1947074560`, output-uul de la Task 1 va fi `4 * * + + + 8`. Rezulta ca vom citi 3 **unsigned short** de la tastatura (((4+1)*8)/16 = 2,5). Presupunem ca vom citi 54998 (**11010110 11010110**), 64041 (**11111010 00101001**) si 42752 (**10100111 00000000**). Practic, primul operand va fi 214, al doilea tot 214, al treilea 250, al patrulea 41, iar al cincilea 167, urmat de bitii de padding.

2. **Executarea instructiunii**: din moment ce avem atat operatiile cat si operanzii, nu ne mai ramane decat sa calculam rezultatul. Calcularea rezultatului se va face in ordinea primirii operatiilor si nu conform prioritatii operatorilor (adica, * nu are precedenta fata de +).

Exemplu:

In cazul in care avem operatiile + - * + si operanzii 1 2 3 4 5 se va valcula 1 + 2 - 3 * 4 + 5 = 5 (1 + 2 = 3 - 3 = 0 * 4 = 0 + 5 = 5)

Pentru acest task, se vor citi de la tastatura instructiunea si operanzii si se va afiza rezultatul:

Exemplu utilizare:

```
./task1

1410859008

3 * * + 4

Numere de introdus = 1

Introduceți număr: 54999

Operanzii: 13 6 13 7

Rezultat: 1021
```

sau

```
./task1

1947074560

4 * * + + 8

Numere de introdus = 3

Introduceți număr: 54998

64041

42752

Operanzii:214 214 250 41 167 0 0 0 0 0

Rezultat: 11449208
```

Precizari:
- rezolvarea acestui exercitiu se va afla in fisierul task2.c

- va recomandam mai intai sa rezolvati taskul 1 si apoi sa faceti copy-paste codului in fisierul task2.c si sa porniti rezolvarea de acolo

- numerele citite de la tastatura sunt considerate fara pozitive, insa rezultatul poate fi negativ asa ca folositi o variabila de tip **int** pentru a salva rezultatul executarii instructiunii.
- afisati doar numarul rezultat, altfel checker-ul nu va lua in considerare testul