Procesor RISC

Pentru că echipa cu care lucrați a dezvoltat o nouă arhitectură de procesor RISC, ați decis să-i ajutați în etapa de dezvoltare prin realizarea unui simulator software pentru procesorul respectiv. Simulatorul va primi, ca și procesorul real, o secvență de instrucțiuni care va trebui executată iar starea finală a datelor va trebui afișată la ieșire. Procesorul are un set de 16 registre numerotate de la 0 la 15 și deocamdată nu are memorie.

Procesorul știe să execute următoarele instrucțiuni, pe care trebuie să le interpreteze și programul vostru:

- **lconst <dst> <val>** – se scrie valoarea <val> în registrul <dst>;
- **add <dst> <src0> <src>** – se adună valorile din registrele <src0> și <src1> și rezultatul se scrie în registrul <dst>;
- **mul <dst> <src0> <src>** – se înmulțesc valorile din registrele <src0> și <src1> și rezultatul se scrie în registrul <dst>;
- **div <dst> <src0> <src>** – se împart valorile din registrele <src0> și <src1> și câtul se scrie în registrul <dst>;
- **print <reg>** - se afișează valoarea din registrul <reg>.

Dacă la execuția unei instrucțiuni de tip **div** împărțitorul este zero, se va afișa fraza **Exception: line <index>**, unde **index** reprezintă a câta instrucțiune nu a putut fi executată, iar programul se va încheia. Toate registrele au inițial valoarea 0.

Cerinţă

Dându-se o secvență de instrucțiuni ca cele de mai sus, executați-le și afișați valorile printate de program.

Date de intrare

Se va citi de la tastatură (fluxul standard de intrare, *stdin*) de pe prima linie un număr **n**, reprezentând numărul de instrucțiuni. Pe următoarele **n** linii se află câte o instrucțiune din cele de mai sus.

Date de ieşire

Programul va afișa la consolă (stream-ul standard de ieşire, *stdout*), valorile printate de program (prin instrucțiuni de tip print), câte una pe linie.

ATENŢIE la respectarea cerinței problemei: afișarea rezultatelor trebuie făcută EXACT în modul în care a fost indicat! Cu alte cuvinte, pe stream-ul standard de ieșire nu se va afișa nimic în plus față de cerința problemei; ca urmare a evaluării automate, orice caracter suplimentar afișat, sau o afișare diferită de cea indicată, duc la un rezultat eronat şi prin urmare la obținerea calificativului „Respins”.

Restricţii şi precizări

1. $0 < n \leq 1000$
2. Registrele pot stoca valori numerice întregi cu semn pe 32 de biți.
3. **Atenţie**: În funcție de limbajul de programare ales, fișierul ce conține codul trebuie să aibă una din extensiile .c, .cpp, .java, sau .m. Editorul web nu va adăuga automat aceste extensii şi lipsa lor duce la imposibilitatea de compilare a programului!

4. **Atenţie**: Fişierul sursă trebuie numit de candidat sub forma: <nume>.<ext> unde nume este numele de familie al candidatului şi extensia este cea aleasă conform punctului anterior. Atenţie la restricţiile impuse de limbajul Java legate de numele clasei şi numele fişierului.

Exemple

Intrare	Ieşire	Explicații

`8`
`lconst 0 10`
`print 0`
`lconst 2 1`
`add 1 0 2`
`mul 2 0 1`
`lconst 1 2`
`div 0 2 1`
`print 0`
	`10`
	`55`
		Sunt opt instrucțiuni în program:
		• **lconst 0 10** va încărca valoarea 10 în registrul 0.
		• **print 0** va afișa valoarea din registrul 0, adică **10**.
		• **lconst 2 1** va încărca valoarea 1 în registrul 2.
		• **add 1 0 2** va aduna valorile din registrele 0 și 2 (10 și 1) și rezultatul 11 va fi scris în registrul 1.
		• **mul 2 1 0** va înmulți valorile din registrele 1 și 0 (11 și 10) și rezultatul 110 va fi stocat în registrul 2.
		• **lconst 1 2** va încărca valoarea 2 în registrul 1.
		• **div 0 2 1** va calcula câtul împărțirii valorilor din registrele 2 și 1 (110 și 2) și rezultatul 55 va fi stocat în registrul 0.
		• **print 0** va afișa conținutul registrului 0: **55**

`7`
`lconst 0 10`
`lconst 1 -1`
`print 0`
`add 0 0 1`
`print 0`
`add 0 0 1`
`print 0`
	`10`
	`9`
	`8`
		Sunt cinci instrucțiuni în program:
		• Primele două instrucțiuni vor încărca valorile 10 și -1 în registrele 0 respectiv 1.
		• print 0 va afișa conținutul registrului 0: **10**
		• add 0 0 1 va aduna valorile din registrele 0 și 1 (10 și -1) și rezultatul 9 se va scrie în registrul 0
		• se repeta ultimele două linii de două ori și deci programul va afișa din **9** și **8**.

`5`
`lconst 0 0`
`lconst 1 1`
`print 0`
`div 3 1 0`
`print 3`
	`0`
	`Exception: line 4`
		Sunt cinci instrucțiuni în program.
		• Primele două instrucțiuni vor încărca valorile 0 și 1 în registrele 0 respectiv 1.
		• **print 0** va afișa conținutul registrului 0: **0**
		• **div 3 1 0** ar trebui să împartă conținutul registrelor 1 și 0, dar pentru că registrul 0 conține 0 se afișează excepție la linia 4 (acesta fiind indexul instrucțiunii care nu a putut fi executată) și programul se oprește.

Timp de lucru: 120 de minute