1. Az Intel Pentium processzor

A stack szerkezetét egy gcc-vel fordított Intel Pentium processzorra generált kódon szemléltetjük. A Windows/XP vagy Linux operációs rendszerek a memória címzésre flat (sík) modellt alkalmaznak. A cím 32 bites, és 0 címtől folytonosan nő. Régebben a MS-DOS rendszerekben ettől eltérő címzést alkalmaztak. A processzorban 8 általános célú (EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP), egy utasítás cím (EIP), és egy flag (EFLAGS) regiszter található. A regiszterek használatát a függelék tartalmazza.

2. Stack szerkezete

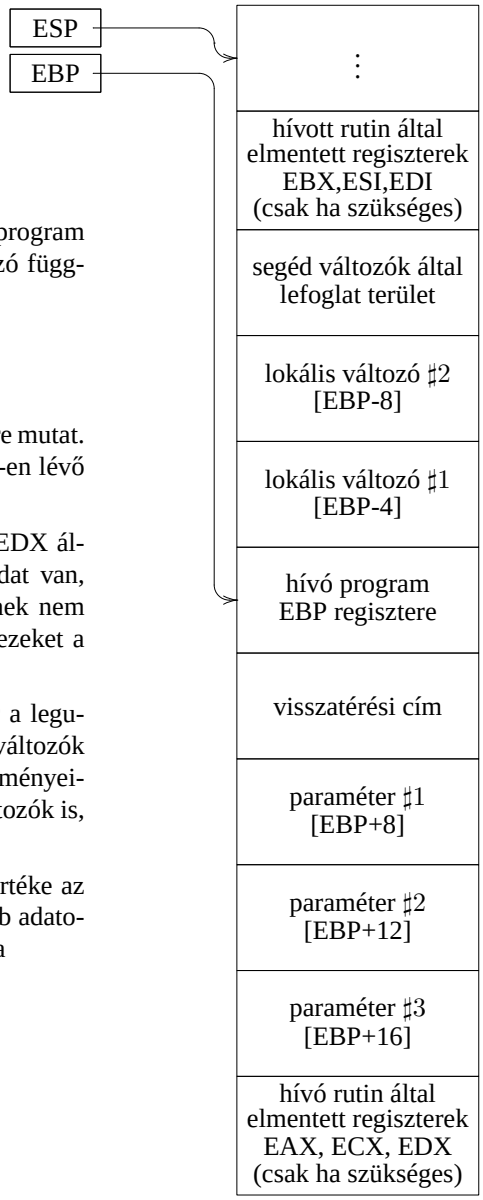
Jobb oldalt egy tipikus stack frame felépítés látható, amint a főprogram meghív egy három paramétert és két lokális változót tartalmazó függvényt. A lap teteje felé helyezkednek el a kisebb címek.
A példában a *hívó* a főprogram `main()`, a *hívott* pedig az:
`int foo(int arg1, int arg2, int arg3);`
A `foo` függvénynek két lokális változója van.

Az ESP (stack pointer) a `foo` függvény stack keretének a tetejére mutat. Az EBP bázis pointeren keresztül éri el a `foo` függvény stack-en lévő paramétereket és lokális adatokat.

A szokás, hogy a hívott függvény elronthatja az EAX, ECX, EDX általános célú regisztereket, ezért ha valamelyikben hasznos adat van, azokat a hívó program elmenti. Másrészt a hívott függvénynek nem szabad elrontani az EBX, ESI és az EDI regisztereket, ezért ezeket a hívott függvénynek kell elmenti.

A függvény paraméterek átadása a stack-en történik. Először a legutolsó paraméter kerül a stack-re, utoljára az első. A lokális változók is a stack-en jönnek létre. Létezhetnek a kifejezések részeredményeinek tárolására, vagy más célra felvett névnélküli temporális változók is, amik szintén a stack-en jönnek létre.

A 4 vagy kevesebb byte-ot tartalmazó függvény visszatérési értéke az EAX regiszteren keresztül kerül átadásra. A 4 byte-nál nagyobb adatokat a fordító egy extra paraméter bevezetésével oldja meg, így a
`x = foo(a, b, c);`
függvény a
`foo(&x, a, b, c);`
formában fordul le.



3. Függvényhívás lépései

Hívó feladata I.

Mielőtt a `main()` meghívná a `foo()` függvényt az ESP és EBP pointer a főprogram stack keretébe mutat.

- A főprogram elmenti EAX, ECX EDX regisztereket.
- A hívó leteszi a hívott függvény paramétereit.

  ```
  a=foo(12,15,18);
  ```

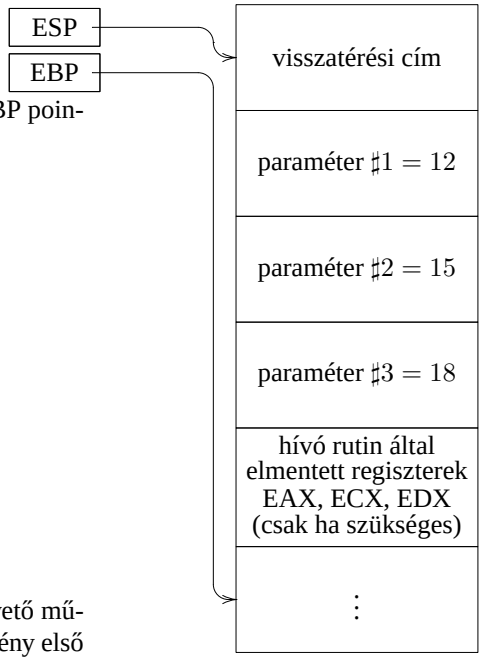
  Assembly utasitások:

  ```
  push dword,18
  push dword,15
  push dword,12
  ```

- a `main()` meghívja a `foo()` függvényt.

  ```
  call foo
  ```

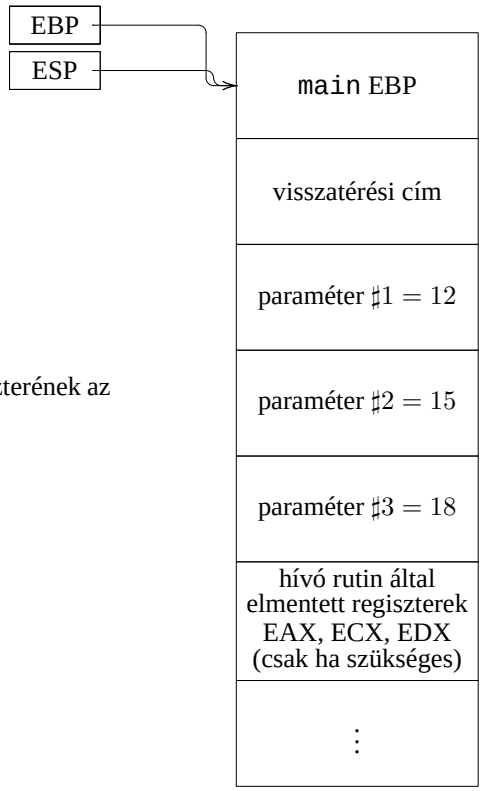
  Ilyenkor az EIP leendő értéké, azaz a `call` utasítást követő művelet címe a stack-re kerül. Az EIP új értéke a `foo` függvény első utasítására mutat.



Hívott feladata I.

A `foo()` fügvény elmenti a `main()` függvény EBP regiszterének az értékét, és az EBP-t a saját stack területre állítja.

```
push ebp
mov ebp,esp
```

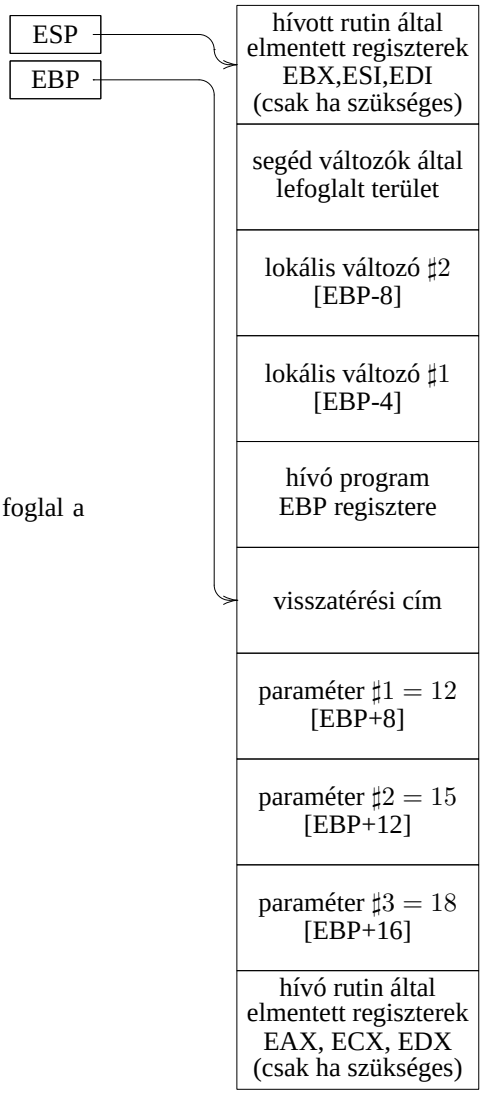


hívott feladata II.

- A hívó a lokális és a temporális adatok számára helyet foglal a stack-en.

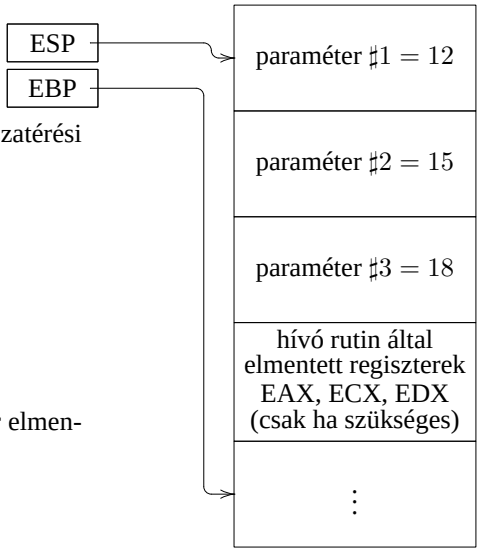
  ```
  sub esp,20
  ```

- Elmenti az EBX, ESI és EDI regisztereket.



4. Visszatérés lépései

Hívott feladata I.



- A `foo()` függvény az EAX regiszterbe beleteszi a visszatérési értéket.
- Visszaállítja az elmentett EBX, ESI, EDI regisztereket.
- ESP stack pointert az EBP által mutatott helyre állítja

  ```
  mov esp,ebp
  pop ebp
  ```

- Visszatér a hívóhoz. Az EIP regiszter felveszi a híváskor elmentett, hívást követő utasítás címét.

  ```
  ret
  ```

Hívó feladata I.

- A `foo()` függvény paramétereit felszabadítja a hívó program. Mivel a példánkban ez 3 darab 4 byte-os adatot jelent, ezért ezt a következő gépi kóddal el lehet intézni:

  ```
  add esp,12
  ```

- EAX-ben lévő visszatérési érték elmentése a megfelelő helyre.
- EAX, ECX, EDX regiszterek visszaállítása.

5. Függelék

Az általános regisztereket az utasítások nem egyformán használják, ezért minden regiszterhez köthető egy fő feladat:

EAX Logikai és aritmetikai műveletek operandusát és eredményét tartalmazza.

EBX Adatok címét tartalmazza.

ECX Számláló string műveletekhez és ciklusok szervezéséhez (gépi kód szinten).

EDX I/O műveletekben használt címet tartalmazza.

ESI Forrás adat címe.

EDI Cél adat címe.

ESP Stack pointer.

EBP Stack címzéshez használt pointer.