Arhitectura sistemelor de calcul

Cod apel intrare / ieșire

Apelul subrutinelor (funcțiilor / procedurilor) este format din 3 etape: cod apel, cod de intrare și cod de ieșire. În momentul apelării unei funcții, sunt necesari anumiți pași pentru ca programul să funcționeze optim. Deși acțiunile depind în funcție de convenția de apel a subrutinei apelate, etapele rămân aceleași.

Codul de apel pregătește și efectuează apelul unei subrutine. Responsabilitățile apelantului constau în:

1. Salvarea resurselor volatile (EAX, ECX, EDX, EFLAGS), presupunând că toți regiștrii își vor modifica valorile în subrutină; acest lucru se poate face cu ajutorul instrucțiunii pushad, care pune pe stivă toți regiștrii, iar pushfd care salvează registrul EFLAGS pe stivă.
2. asigurarea faptului că ESP este aliniat, DF = 0
3. pregătirea argumentelor pe stivă conform convenției folosind instrucțiunea push dword parametru.
4. efectuarea apelului cu adresă de revenire (call).

exemplu: segment code use32:

start:

```
mov ecx, 10

xor eax, eax

reluta:
    push eax
    push ecx                    } salvarea resurselor volatile

    push eax
    push dword format           } pregatirea parametrilor
    call [printf]               } efectuarea apelului
    add esp, 2 * 4

    pop ecx
    pop eax                     } restaurarea resurselor volatile
loop reluta
```

Codul de intrare este codul scris la începutul unei subrutine. Responsabilitățile apelantului constau în:

1. crearea unui cadru de stivă nou folosind seria de instrucțiuni `push ebp` și `mov ebp, esp` (EBP este folosit ca un punct de referință pentru a accesa variabilele locale și parametrii unei funcții)
2. alocarea de spațiu pentru variabilele locale prin scăderea numărului de octeți necesari din ESP.
3. Salvarea unei copii a resurselor nevolatile modificate (push)

exemplu:

```
_suma_numere:
    push ebp
    mov ebp, esp          } crearea cadrului de stivă
    mov eax, [ebp + 8]
    mov ebx, [ebp + 12]
    add eax, ebx
    mov esp, ebp
    pop ebp
    ret
```

Codul de ieșire este codul scris la finalul unei subrutine apelate. Responsabilitățile apelantului constau în:

1. eliberarea variabilelor locale ale subrutinei folosind instrucțiunea mov esp, ebp
2. eliberarea cadrului de stivă folosind instrucțiunea pop ebp.
3. revenire din subprogram și eliberarea de pe stivă parametrii: dacă este de tip CDECL se folosește ret, iar în subprogramul apelant se scrie add esp, dimensiune_argumente, iar dacă e de tip STDCALL se scrie ret dimensiune_argumente.

exemplu:

```
_suma_numere:
    push ebp
    mov ebp, esp
    mov eax, [ebp+8]
    mov ebx, [ebp+12]
    add eax, ebx
    mov esp, ebp
    pop ebp
    ret
```

} cod de ieșire

Cadrul de stivă este o structură de date stocată în stivă, de dimensiune fixă, pentru o subrutină dată, care conține parametrii pregătiți în codul de apel, adresa de revenire, copii ale surselor nevolatile folosite de acea subrutină.