Assembler

Peter Švec
peter_svec@stuba.sk

Assembler

Tri základné koncepty:

- Inštrukcie
  - Manipulácia s dátami
  - Kontrola toku programu
  - Systémové volania
- Registre
  - Ukladanie dočasných dát
- Pamäť
  - Samotné inštrukcie
  - Zásobník

Registre

Malé (8 b na amd64) a rýchle

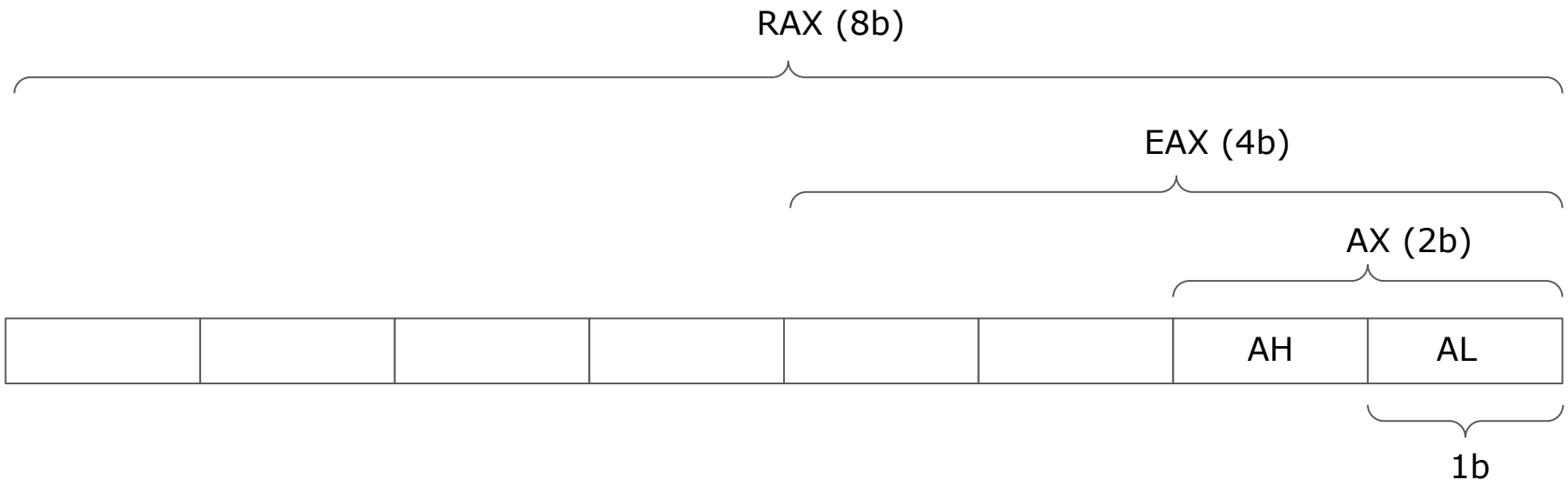
Všeobecné registre:

- *8086*: AX, BX, CX, DX, **SP**, **BP**, SI, DI
- *x86*: EAX, EBX, ECX, EDX, **ESP**, **EBP**, ESI, EDI
- *amd64*: RAX, RBX, RCX, RDX, **RSP**, **RBP**, RSI, RDI, R8, R9, R10, R11, R12, R13, R14, R15

Adresa ďalšej inštrukcie, ktorá sa má vykonať:

- **IP** (8086), **EIP** (x86), **RIP** (amd64)

Registre



Pozor: prístup k E.., vynuluje zvyšok registra

RAX = FF FF FF FF FF FF FF FF
EAX = AA AA AA AA
RAX = 00 00 00 00 AA AA AA AA

Inštrukcie

Všeobecná forma:

INŠTRUKCIA OPERAND, OPERAND, ...

INŠTRUKCIA - čo sa má vykonať

OPERAND - nad akými dátami sa to má vykonať

```
    mov rcx, rax
    add rcx, 5
    cmp rcx, rbx
    je nejake_navestie
nejake_navestie:
    inc rcx
```

```
mov rax, rbx
mov rax, [rbx+4]
add rax, rbx
mul rsi
inc rax
inc [rax]
```

Tok programu

Skoky prebiehajú na základe registra eflags(x86), rflags(amd64)

Hodnoty v rflags sa nastavujú pri:

- Aritmetických operáciach (**add**, **mul**, **div**, **sub**,...)
- Inštrukcia cmp -> **cmp** rax, rbx (rax - rbx)
- Inštrukcia test -> **test** rax, rax (rax & rax)

rflags:

ZF - zero flag

OF - overflow flag

SF - signed flag

CF - carry flag

cmp rax, rbx
je navestie
; ZF = 1

cmp rax, rbx
ja navestie
; CF = 0 & ZF = 0

jmp navestie

Ďalšie možné inštrukcie pre skoky:
jne, **jg**, **jl**, **jle**, **jge**,...

Systémové volania

- Interakcia s operačným systémom
- open, read, write, fork, execve…
- Inštrukcia **syscall**
- exit(42):

```
mov rax, 60
mov rdi, 42
syscall

; navratova hodnota v RAX
mov rcx, rax
```

https://blog.rchapman.org/posts/Linux_System_Call_Table_for_x86_64/

Zásobník (stack)

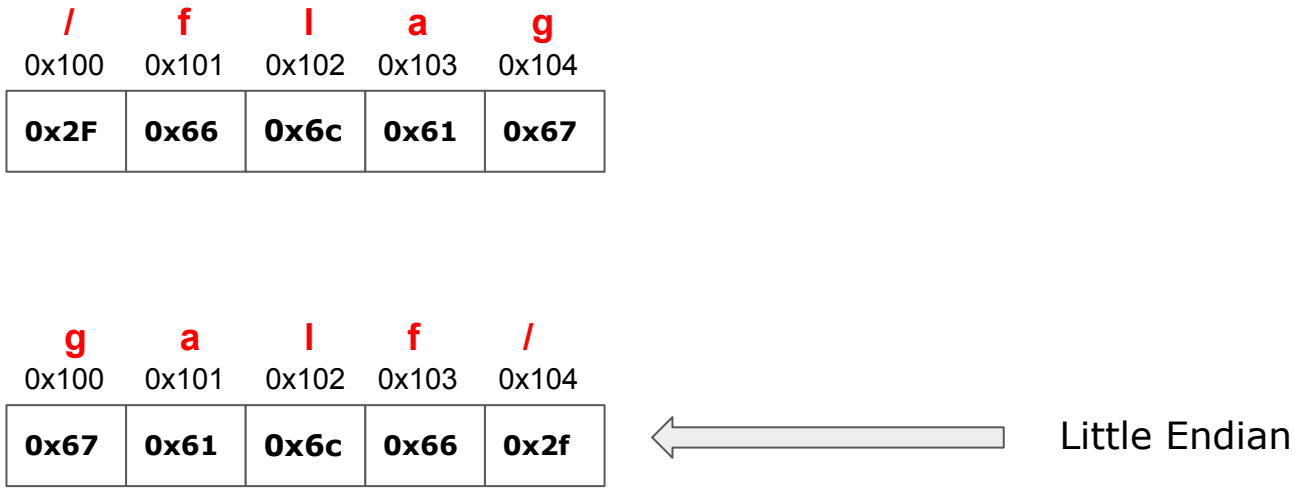
- Zasobníkový rámec (ZS) pre volania funkcií
- Obsahuje:
  - Uložený ZS predchádzajúceho volania
  - Lokálne premenné pre funkciu
  - Návratová adresa (návrat z funkcie - inštrukcia **RET**)
- Registre ovládajúce zásobník (x64): **RSP** (vrch zásobníka), **RBP** (spodok zásobníka)
- Práca so zásobníkom (x64): **PUSH**, **POP**

```
push rax
push 0xff
```

```
pop rax
pop [rcx]
```

Pamäť (endianita)

Dáta (vo väčšine systémoch) sú uložené opačne - Little Endian



Ďalšie inštrukcie

- Logické operácie:
  - XOR, AND, OR, NEG,...
- Bitové posuny, rotácie:
  - SHR, SHL, ROL, ROR,...
- Volanie funkcie:
  - CALL, RET
- Načítanie adresy:
  - LEA (Load Effective Address)
  - LEA RAX, [EBX + 8 * RAX] (EBX = adresa zaciatku array...array[RAX])
- Žiadna operácia:
  - NOP

Ďalšie zdroje

- Rappel
  - https://github.com/yrp604/rappel
- Inštrukcie
  - https://www.felixcloutier.com/x86/