Vývojové prostředí pro asemblery

Ing. Dominika Regéciová
Výzkumná skupina formálních modelů

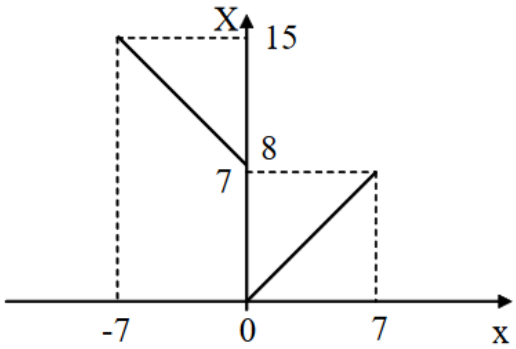
Vysoké učení technické v Brně, Fakulta informačních technologií
Božetěchova 1/2, 612 66 Brno - Královo Pole

iregeciova@fit.vutbr.cz

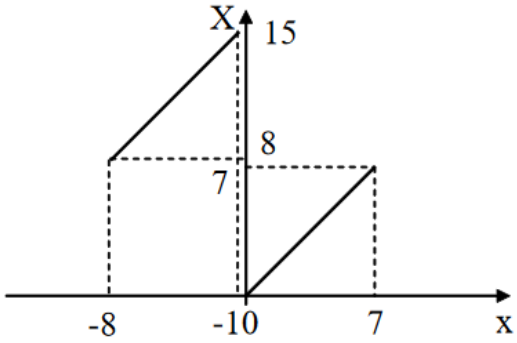
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Příklad na začátek

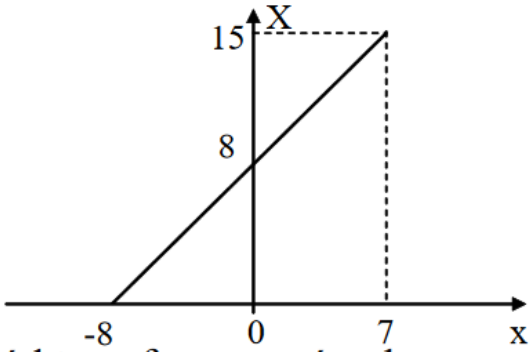
Převeďte číslo −111 do přímého, doplňkového kódu a kódu transformované nuly (na 8 bitů).



a) Přímý kód x ∈ <-7, 7>



b) Doplňkový kód x ∈ <-8, 7>



c) Kód transformované nuly x ∈ <-8, 7>

Jazyk symbolických instrukcí: Instrukce

- Používáme symbolické instrukce - označení kódového příkazu pro elementární operace procesoru (sečtení dvou čísel, instrukce skoku, . . .). Posloupnost těchto instrukcí, strojový kód, se poté překládá do binární podoby.

- Instrukce může být až dva operandy
  - `instukce`
  - `instukce DEST`
  - `instukce DEST, SOURCE`

- Operandem může být
  - Registr `(AL, AH, AX, EAX, BL, ...)`
  - Adresa v paměti `(sMessage, ...)`
  - Konstanta `(65, 0b0111100, 0o101, 0x41, 'A', ...)`

- **Víte, proč nejdříve uvádíme cílový registr?**

Jazyk symbolických instrukcí: Registry

- Pracujeme přímo s registry procesoru - malé úložiště dat, které ale mají rychlé operace čtení a zápisu
- Registry jsou umístěny v procesoru - jsou malé a je jich obvykle málo (16 v 64-bit procesoru pro obecné použití)
- Operace s nimi - čtení a zápis - jsou velmi rychlé
- Registry pro obecné použití:
  - **Datové**
  - **Ukazatelé**
  - **Indexové**

Jazyk symbolických instrukcí: Registry II

- **Datové**
  - *EAX* - aritmetické operace, vstup/výstup
  - *EBX* - adresování
  - *ECX* - čítač v cyklech
  - *EDX* - vstup/výstup, aritmetické operace mul, div
- **Ukazatelé**
  - *EIP* - instruction pointer, ukazatel na následující instrukci
  - *ESP* - offset v rámci programového zásobníku
  - *EBP* - pomáhá při referencování proměnných předávaných do subrutiny
- **Indexové**
  - *ESI* - zdrojový index pro operaci s řetězci
  - *EDI* - cílový index pro operace s řetězci

Registry

31 … 16	15 … 8	7 … 0
A … Accumulator	AH	AL
	AX	
EAX		
B … Base	BH	BL
	BX	
EBX		
C … Counter	CH	CL
	CX	
ECX		
D … Data	DH	DL
	DX	
EDX		
Code Segment	CS	
Data Segment	DS	
Extra Data Seg.	ES	
Stack Segment	SS	

31 … 16	15 … 0
SP … Stack Pointer	SP
ESP	
BP … Base Pointer	BP
EBP	
SI … Source Index	SI
ESI	
DI … Destination Index	DI
EDI	
IP … Instruction Pointer	IP
EIP	
registr příznaků	FLAGS
EFLAGS	

H … High byte, **L** … Low byte

E … Extended (= 32bitový režim)

další segmentové registry – FS, GS

Vývojové prostředí pro asemblery

- Budeme pracovat v operačním systému MS Windows
- Doporučené prostředí: SASM
- Alternativně: NASM a GoLink
- Informace pro Linux naleznete buď na wiki stránce, nebo na stránce kolegy Budiského: http://www.fit.vutbr.cz/ ibudisky/isu/2019/

SASM

- Vše potřebné naleznete v archivu `SASM-ISU-2018.zip` na Wiki stránkách předmětu
- Jednoduché grafické vývojové prostředí, umožňuje psaní kódu, jeho překlad a ladění
- Není nutná instalace, stačí spustit `sasm.exe` soubor

Debugování v SASM

```
%include "rw32-2018.inc"

section .data
    ; zde budou vase data

section .text
_main:
    mov ebp, esp
    mov al, 0xf1
    mov ah, 0xff
    xor eax, eax
    ret
```

Překlad prostřednictvím příkazové řádky

- stáhněte si `isu-tools-2018.zip` na Wiki stánkách předmětu
- Rozbalte si archiv na disku P do adresáře ISU
- Spusttě příkazovou řádku a přes příkaz `cd` změňte aktuální adresář na místo, kde máte uložené isu-tools
- Příkazem `dir` si můžete ověřit, že jste ve správném adresáři
- Přeložte a spustě ukázkový program pomocí příkazu:
  `run helloworld`

Dobré vědět . . .

Na příští cvičení se vám může hodit přehled základních instrukcí: http://www.jegerlehner.ch/intel/IntelCodeTable.pdf