



VILNIAUS GEDIMINO
TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

Virtuali mašina

LEKT. DR. PAVEL STEFANOVIČ

5 laboratorinis darbas (1)

- Pasirinktoje programavimo kalboje realizuokite virtualią mašiną, kuri:
 - turi 16 bendros paskirties registrų *R0-R15* (8 bitų pločio);
 - turi 8 bitų programos adresavimą (instrukcijų skaitiklis yra 8 bitų pločio);
 - turi 256 baitų programų atmintį;
 - turi „vėliavų” (*angl.* flag) registrą – tik *IN* komandai.
- Užduočiai duota: virtualios mašinos specifikacija:
 - programos dvejetainis failas (*decryptor.bin*);
 - tekstinis įėjimo failas (*q1_encr.txt* – užšifruoti duomenys).

5 laboratorinis darbas (2)

- Užduotis laikoma atlikta, kai mašina, vykdydama duotą programą *decryptor.bin* ir skaitydama įvesties duomenis (*q1_encr.txt*), iššifruos tekstą (anglų mokslininko sentenciją apie kompiuterių vystymąsi, anglų kalba) išvesdama jį į išėjimo įrenginį (pvz.: konsolės langą).
- Galite naudoti bet kokias priemones, bet C/C++ rekomenduotina. Virtualios mašinos specifikacija bus pateikta C kalba.

P. S. kad būtų aiškiau kas slepiasi duotoje programoje, pasinaudokite bet kuria „hexdump“ programėle, pvz.: <https://hexed.it/>

Virtualios mašinos specifikacija (1)

- Komandų struktūroje yra 2 tipų komandos:
 - **1 tipas (komanda su 2 parametrais).** Šios komandos skirtos darbui su registrais, jų formatas:

8 bitų komandos kodas	
Bitai 7-4 , 2-as param. (registas)	Bitai 3-0 , 1-as param. (registas)

- Kai kurios komandos (*LSL*, *INC*) neturi 2-o registro, tuo atveju bitai 7-4 ignoruojami.

Virtualios mašinos specifikacija (2)

- **2-o tipo komandos (su 1 parametru):** 1 baido komandos kodas ir 1 baido konstanta (*MOVC*) arba šuolio (*JZ*, *JNZ*, *JFE*) atstumas.

8 bitų komandos kodas
8 bitų konstanta arba šuolio nuotolis

Virtualios mašinos komandos (1)

Komanda	Baito kodas	Komentaras
INC Rx	0x01	Registrą x padidina vienetu
DEC Rx	0x02	Registrą x sumažina vienetu
MOV Rx, Ry	0x03	Kopijuoja registrą Ry į Rx
MOVC «baito konstanta»	0x04	Kopijuoja baito konstantą į registrą R0. 2-o tipo komanda
LSL Rx	0x05	Registro Rx postūmis į kairę per 1 bitą
LSR Rx	0x06	Registro Rx postūmis į dešinę per 1 bitą
JMP addr	0x07	Šuolis santykinio adresu pridedant konstantą (su ženklų) «addr» prie komandų skaitiklio. Visi šuoliai yra 2-o tipo komandos
JZ addr	0x08	Kaip ir JMP, tik jei yra nulinio požymis (flag). Šiame darbe nenaudojama
JNZ addr	0x09	Kaip ir JMP, tik jei nėra nulinio požymio. Šiame darbe nenaudojama
JFE	0x0A	Kaip ir JMP, tik jei yra failo pabaigos požymis – žr. IN komandą

Virtualios mašinos komandos (2)

RET	0x0B	VM baigia darbą
ADD Rx, Ry	0x0C	Prideda registro y turinį prie registro x ir palieka rezultatą registre x
SUB Rx,Ry	0x0D	Atima registro y turinį iš registro x ir palieka rezultatą registre x
XOR Rx,Ry	0x0E	«Išimtinio arba» (XOR) operacija kiekvienam bitui atskirai. Rezultatas paliekamas Rx
OR Reg1,Reg2	0x0F	«Arba» (OR) operacija kiekvienam bitui atskirai. Rezultatas paliekamas Rx
IN Rx	0x10	Skaito vieną baitą iš duomenų failo ir nustato failo pabaigos požymį jei pasiekta failo pabaiga.
OUT Rx	0x11	Registro x turinį išveda į ekraną / failą

Galimas realizacijos pavyzdys C kalboje: (1)

- Registrai realizuoti kaip *char* masyvas: *unsigned char regs[16];*
- Programų atmintis realizuota kaip *char* tipo masyvas: *char prog_mem[256];*
- Vėliavų registrą galima realizuoti kaip paprastą *int* tipo kintamąjį. VM yra labai paprasta, joje nėra komandų, kaip pvz.: *INT*, kuri vėliavų registrą saugotų steke (stekas šioje VM irgi neegzistuoja): *int ieof_flag;*

Galimas realizacijos pavyzdys C kalboje: (2)

- Programa nuskaitoma į programų atmintį elementarių *stdio.h* funkcijų pagalba. Kadangi mašinos dvejetainėje programoje yra *ASCII* valymo kodų ($<0x20$), skaitant failą jie interpretuojami. Todėl failą reikia atidaryti dvejetainiame režime (“*rb*”):

```
FILE* fp = fopen( <...programos failas...>. , “rb” );
if( fp == NULL ) {
    printf( “Neiseina atidaryti programos failo\n” );
    exit(1);
} int i;
for( i = 0; i < sizeof(prog_mem); i++ ) {
    fread( prog_mem+i, 1,1, fp );
    if( feof( fp ) ) break;
} fclose( fp );
```

Galimas realizacijos pavyzdys C kalboje: (3)

- Apibrėžkite VM kaip struktūrą:

```
typedef struct {  
    char code;  
    char cop1;  
} VMCommand;
```

- Komandų registras tokiu atveju yra paprasta nuoroda (*pointer*), kurios tipas – *VMCommand*. VM pradedant darbą jis rodys į pirmą komandų atminties baitą. Su kiekviena komanda nuoroda yra didinama vienetu ir dėl nuorodų aritmetikos realiai „šokinėja“ per 2 baitus:

```
VMCommand* r_pc;
```

```
r_pc = (VMCommand*) prog_mem;
```

Galimas realizacijos pavyzdys C kalboje: (4)

- Prieš kiekvienos komandos vykdymą galite iš anksto paskaičiuoti komandoje naudojamų registrų numerius. Kintamieji laikinam registrų numerių skaičiavimui: *int iReg1, iReg2;*
- Registrai skaičiuojami turint galvoje kad programų skaitiklis jau rodo į komandą, kurią VM šiuo metu vykdo:

iReg1 = r_pc->cop1 & 0x0F;

iReg2 = r_pc->(cop1 & 0xF0) >> 4 ; // arba dalinti iš 16-kos, kas yra tas pats

- Po kiekvienos komandos komandų skaitiklis yra didinamas vienetu ir dėl nuorodų aritmetikos, C kalboje „šokinėja“ per 2 baitus:

r_pc++; // r_pc jau rodo į sekančią komandą

Galimas realizacijos pavyzdys C kalboje: (5)

- Komanda *IN* skiriasi nuo komandos *IN* procesoriuje: ji skaito 1 baitą iš duomenų failo ir kopijuoja jį į registrą, nurodytą komandoje. Pvz.: *IN R1* skaito baitą į *regs[1]*. Duomenų failo skaitymui pasiruošiama prieš komandų vykdymą, naudojama ta pati kaip ir programų skaitymo atveju *fopen* funkcija:

```
FILE* fdata;  
fdata = fopen( <... duomenų failas ...> , "r" );  
if( fdata == NULL ) {  
    printf( "Neiseina atidaryti duomenų failo \n" );  
    exit(1);  
}  
3  
// Duomenų failas atidarytas, failo nuoroda sukurta, failas atidarytas ir "laukia" skaitymo  
IN komandos pavyzdys. Atkreipkite dėmesį į feof naudojimą:  
fread( regs[ iReg1], 1, 1, fdata );  
if( feof( fdata ) ) {  
    ieof_flag = 1;  
} else {  
    ieof_flag = 0;  
}
```

ieof_flag bus naudojamas JFE komandos atveju

Galimas realizacijos pavyzdys C kalboje: (6)

- Pastaba dėl *JMP* šuolio skaičiavimo C kalboje: jose po komandos kodo seka šuolio adresas. Reikšmė yra baito (*char* tipo) su ženklu, šuolio atstumas skaičiuojamas baitais. Tačiau programų skaitiklis šiuo atveju yra *VMCommand** (2 baitų nuorodos tipo). Pridedant tam tikrą baitų skaičių dėl nuorodų aritmetikos komandų skaitiklis judės visada per 2 baitus. Vienas iš galimų šios problemos sprendimų: *r_pc* yra konvertuojamas į *char* tipo nuorodą ir tada vyksta šuolio konstantos sumavimas. Gautas adresas vėl konvertuojamas į *VMCommand* tipo nuorodą:

$$r_pc = (VMCommand*) ((char*) r_pc + r_pc->cop1);$$

Virtualios mašinos struktūrinė schema

