Západočeská univerzita v Plzni
Fakulta aplikovaných věd

KIV/POT

Dokumentace semestrální práce

Změna malého písmen na větší a vice versa na základě podmínek

Jan Novák
A17B0309P

Zadání

Program vezme vstup ze simulovaného vstupu a poté u každého slova změní první písmeno toho slova z velkého na malé a z malého na velké, pokud je dané písmeno na začátku slova.

Obsah

Popis programu

Tabulka proměnných

Název	Adresa	Popis
stck	0x00FF40CC	začátek zásobníku
buffer	0x00FF4000	Začátek datového bloku
textt	0x00FF4068	Ukazatel na začátek datového bloku

Procedury

_start

Vstupní bod aplikace

Adresa: 0x00000490

Použité registry:

- ER7 - pointer na stack
- ER2 - pointer na textt později na začátek zadaného text
- R0 - pointer na simulovaný vstup
- ER1 - adresa parametrického bloku
- E3 - bit pro porovnání
- R4L - hodnota indikace konce zadávaného textu
- R4H - hodnota mezery pro oddělení slov
- E4 - „boolean“ indikace nalezení mezery
- R5L - dolní hranice pro velká písmena
- R5H - horní hranice pro velká písmena
- R6L - dolní hranice pro malá písmena
- R6H - horní hranice pro malá písmena

Hlavní procedura aplikace ve které se připraví všechny registry a spustí hlavní výkonnou proceduru cycle

cycle

hlavní výkonná procedura programu

Adresa: 0x00046C

Použité registry:

ER2 - pointer na aktuální znak

R3L - aktuální znak

R4L - dolní hranice znaku

R4H - horní hranice znaku

E4 - boolean jestli byla nalezena mezera

E3 - hodnota „true“

Procedura cycle je hlavní výkonnou částí programu, zde se provádějí úvodní podmínky, test na mezeru a přeskočení znaku.

clear

čištění registrů

Adresa: 0x00000400

Použité registry:

ER0 – ER6 - všechny tyto registry se nastaví na nulu(0x000000)

Procedůra čištění registrů na začátku programu. Nastaví na nulu všechny registry až na ER7, kde je stack pointer

print

vypsání textu

Adresa: 0x0000042C

Použité registry:

R0 - hodnota pro PUTS

ER1 - adresa param. bloku pro ER1

Procedura vypíše obsah paměti na simulovaný výstup

Návěští

check_letters

první sada podmínek rozdělování

Adresa: 0x00000464

Použité registry:

R3L - aktuální zpracovávaný znak

R5L - dolní hranice velkých písmen

R6L - dolní hranice malých písmen

Návěští slouží k rozdělení velkých a malých písmen podle dolní hranice, kde v dalších návěštích se kontroluje horní

letter_low

Podmínka horní hranice malých znaků

Adresa: 0x00000454

Použité registry:

R3L - aktuální znak

R6H - hodnota horní hranice malých znaků

Návěští slouží jako podmínka horní hranice malých znaků

change_low

Změna malého znaku na velký

Adresa: 0x0000043C

Použité registry:

R3L - hodnota znaku

ER2 - pointer na aktuální znak v paměti

E4 - boolean pro mezeru

Návěští změní malý znak na velký pomocí odečtení čísla 32 od registru R3L

letter_high

Podmínka horní hranice velkých znaků

Adresa: 0x0000045C

Použité registry:

R3L - hodnota znaku

R5H - hodnota horní hranice velkých znaků

Návěští slouží jako podmínka horní hranice velkých znaků

change_high

Změna velkého znaku na malý

Adresa: 0x00000448

Použité registry:

- R3L - hodnota znaku
- ER2 - pointer na aktuální znak v paměti
- E4 - boolean pro mezeru

Návěští změní velký znak na malý za pomoci přičtení čísla 32 k registru R3L

end_change

Ukončení jednoho cyklu zpracování znaku

Adresa: 0x0000047A

Použité registry:

- ER2 - pointer na aktuální znak v paměti
- E4 - boolean pro mezeru

Ukončení cyklu, nastaví se registr E4 na 0 a dojde k posunutí pointeru na znak v paměti o jeden byte.

end_change

Ukončení jednoho cyklu zpracování znaku

Adresa: 0x0000047A

Použité registry:

- ER2 - pointer na aktuální znak v paměti
- E4 - boolean pro mezeru

Ukončení cyklu, nastaví se registr E4 na 0 a dojde k posunutí pointeru na znak v paměti o jeden byte.

end_set_space

Ukončení jednoho cyklu zpracování znaku a nastavení mezery

Adresa: 0x00000484

Použité registry:

- ER2 - pointer na aktuální znak v paměti
- E4 - boolean pro mezeru

Ukončení cyklu, nastavení E4 na 1 jako detekce mezery a dojde k posunutí pointeru na znak v paměti o jeden byte.

Ukázka programu ve vyšším jazyku

Než jsem začal psát ASM, tak jsem si napsal program, který dělá stejnou funkčnost jako výsledný ASM program abych se mohl zamyslet nad algoritmem.

```
class Program
{
    private static bool space = true;
    private static string input = string.Empty;
    private static string output = string.Empty;


    static void Main(string[] args)
    {
        input = Console.ReadLine();
        output = string.Empty;


        foreach (var ch in input)
        {

            int unicodeCharacter = Convert.ToInt32(ch);

            if (space)
            {
                if (unicodeCharacter >= 60 && unicodeCharacter <= 90)
                {
                    WriteToOutput(unicodeCharacter + 32);

                }
                else if (unicodeCharacter >= 97 && unicodeCharacter <= 122)
                {
                    WriteToOutput(unicodeCharacter - 32);
                }
                space = false;
            }
            else if (unicodeCharacter == 32)
            {
                space = true;
                WriteToOutput(unicodeCharacter);
            }
            else
            {
                WriteToOutput(unicodeCharacter);
            }
        }

        Console.WriteLine(output);
        Console.ReadKey();
    }

    private static void WriteToOutput(int unicode)
    {
        output += ((char) unicode).ToString();
    }
}
```

Obsluha aplikace

Po zapnutí programu zapíšete vstup do simulovaného vstupu. A Program vám následně vrátí zpracovaný výstup např. „Tohle je vstup“ → „tohle Je Vstup“.