Semestrální práce z počítačové techniky

Jan Novák
A11B0442P
nowitz@students.zcu.cz

ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Fakulta aplikovaných věd Západočeské univerzity v Plzni

Jan Novák-A11B0442P

Obsah

1) Zadání 1

2) Řešení 1

a) Specifikace **1**

b) Analýza **1**

c) Implementace **1**

d) Design **2**

e) Tabulka definovaných proměnných **2**

3) Ukázka 3

4) Závěr 3

1) Zadání

V zadaném řetězci program určí počty slov se stejnou délkou. Počty uloží do paměti v desítkové soustavě.

Zadání samostatných prací budou jednoduché programy k naprogramování v JSA. Vstupní data budou zadávána pomocí konzole. Výstupní data budou zobrazena na konzoli. V referátu proto musí být (kromě jiného) uvedeno, na jakých adresách paměti se vkládají vstupní data a na jakých adresách je uložen výsledek.

Každý program musí obsahovat alespoň jeden kódový a jeden datový segment. Dále musí být řešení každé z úloh navrženo tak, aby obsahovalo alespoň jeden podprogram, který je v programu volán. Proto musí být správně inicializován a použit zásobník.

2) Řešení

a) Specifikace

Program jsem si specifikoval, že jako vstupní řetězec bere veškeré znaky a slova jsou oddělená mezerou. Dále výpis na konzoli je pouze pro slova od velikosti 1 až 9. Při zadávání vstupu se zadává celý řetězec, ve kterém chci počítat počet slov o stejné délce. Tento řetězec můsí mít maximálně 99 znaků a je ukončen klávesou enter. Velikost pole pro uchování výsledků je 20 bytů a velikost pole pro převedení hodnot do (ascii dec) je 40 bytů. Jinak u tohoto zadání není potřebná nějaká větší (bližší) specifikace. Vše je napsané na úvodu dokumentu v zadání.

b) Analýza

U této úlohy je důležité rozhodnout, jak vypisovat data na konzoli, která jsou v hexa kódu. Tento problém má více řešení. Buď k hodnotě přičíst 0x30 abych dostal například hodnotu 1 v ascii kódu (dec 1 = ascii 0x31), což by ale vedlo pouze k řešení výpisu čísel 0 až 9. Dále se tu pokládá možnost tzv. dělením 10.

c) Implementace

V paměti si alokuji místo o velikosti 20 a 40 bytů, do něj budu ukládat počty slov o určitých délkách a následně převedené hodnoty do (ascii dec). Dále si nuluji veškeré registry a jednotlivé pole, abych měl jistotu, že nezůstala v žádném nežádoucí hodnota. Provádím ještě úpravu textu txt2 , aby vše bylo připraveno pro správný chod programu. Tyto úpravy slouží hlavně pro opětovný chod programu.

Pak pomocí zásobníku (stack pointeru) vypisuji hlášku pro zadání řetězce a následně zadaný řetězec ukládám do paměti (buffer). Poté opět nuluji veškeré registry pro zajištění správného chodu programu.

Vezmu adresu bufferu, vložím si jí do registru a načtu první znak a spustím cyklus. Ten běží, dokud nenarazí na mezeru, konec řetězce nebo znak NL(new line) a přičítá si délku slova do registru (ER1). Když narazí na mezeru, konec řetězce nebo znak NL skočí na návěští uloz, kde se vezme adresa pole, přičte se k ní 1 a následně se i odečte hodnota počítadla (ER1) o jedničku. Tím dostávám požadovanou adresu pro umístění hodnoty, ve kterém zvýšíme hodnotu o jedna. To provádím tak, že původní hodnotu registru ER2 uložíme do registru E3L, přičtu k němu jedničku a uložíme zpět do paměti. Následně se kontroluje, zda poslední znak není konec řetězce. Pokud ano, program skočí na návěští prevod, pokud ne, vynuluje se počítadlo délky slova a načte se další znak řetězce a skočí se zpět do smyčky.

Převod provádím pomocí dělení 10 se zbytkem. Převedu obsah registru R1L (hex) na R2 (ascii dec), pomocí registru R3L a dělením deseti (divxu.b). Nově převedená čísla si ukládám do pole_vystup.

Výpis provádím pomocí načtení prvních dvou uložených hodnoty z pole_vystup. Pomocí registru R6L, je postupně vkládám do parametrického bloku par3. Poté dochází k výpisu textu, který určuje velikost slov v řetězci. Pod tento výpis se vypisuje konkrétní hodnota počtu slov k předem vypsané délce. Tento výpis je zacyklený, než dojde k zastavovací podmínce, která ukončí následně i celý program.

d) Design

Program je naprogramován v programovacím jazyce symbolických adres. Využil jsem vývojové prostředí HEW (High performance Embedded Workshop). Součástí vývojového prostředí je i simulátor několika procesorů včetně procesoru H8S/2600. Dále využíváme nástroje GNUH8 v12.02 Windows Tool Chain (ELF). Můj kód je přenositelný na jakýkoli počítač a dále rozšiřitelný.

e) Pozice proměnných v paměti

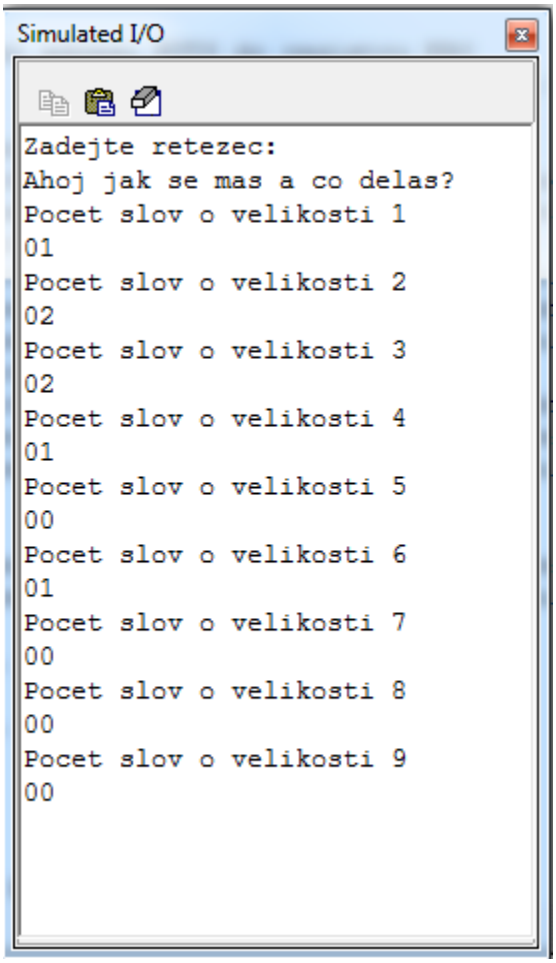
Tabulka definovaných proměnných a jejich umístění v paměti:

Název	Adresa (umístění)	Význam
pole_vystup	0xFF40B8	pole pro uchování převedených výsledků
pole	0xFF40A4	pole pro uchování výsledků
txt1	0xFF4000	text pro výzvu zadaní řetězce
txt2	0xFF4012	text pro výpis velikosti slov
buffer	0xFF402C	buffer pro načtení zadané hodnoty z konzole
vystup	0xFF4090	pole pro výstupní text
par1	0xFF4035	parametrický blok pro txt1
par2	0xFF4099	parametrický blok pro buffer
par3	0xFF409D	parametrický blok pro txt2
par4	0xFF40A1	parametrický blok pro výstup
stck	0xFF4114	prostor pro zásobník

UŽIVATELSKÁ DOKUMENTACE

Maximální délka řetězce může být 99 znaků. Pro ukončení zadaného řetězce je zapotřebí zmáčknout klávesu enter. Dále uživatelská dokumentace není u tohoto programu potřebná, protože po spuštění programu je uživatel pomocí výpisů na konzoli veden.

3) Ukázka



4) Závěr

Program splňuje zadání a nebyly zjištěny žádné chyby v návrhu. Výstup je ukládán do paměti a následně vypsán na konzoli. Program by měl byt schopen více běhů. Program by se jistě dal řešit efektivněji, například zvolením jiného algoritmu nebo efektivnějším využitím registrů.

Můj subjektivní závěr je, že tato práce byla zajímavá a dala mi nové zkušenosti v programovacím jazyce symbolických adres. Se svojí prací spokojen.