Západočeská univerzita v Plzni
Fakulta aplikovaných věd

KIV/UPA

Dokumentace semestrální práce

Fibonnaciho posloupnost

Jakub Vítek
A16B0165P

1. Zadání

Program vypočítá a vypíše požadovaný počet členů fibonacciho posloupnosti. Případný vstup je zadáván z klávesnice, výstup na obrazovku ve vhodném formátu se základními informacemi pro uživatele. Minimálně lze vyžádat dva členy k výpisu a maximálně lze vyžádat tolik členů, kolik dovoluje velikost registru. Program by mel obsahovat alespoň jednu funkci.

2. Obsah

3. Popis programu

3.1. Tabulka použitých proměnných

Název	Adresa	Popis
fix	0x10010000	Ukazatel na text, který se bude vypisovat ve chvíli, kdy jedna z hodnot fibonacciho posloupnosti způsobí přetečení registru.
msg	0x1001004C	Ukazatel na text, který se bude vypisovat na začátku programu - otázka pro uživatele kolik členů posloupnosti chce vypsat
lim	0x10010086	Ukazatel na text, ktery se bude vypisovat v případě, že počet členů posloupnosti zadaný uživatelem je menší než dva
nl	0x100100D4	Ukazatel na text, který vypisuje v případě nutnosti přechodu na nový řádek v konzoli - line feed.

3.2. Seznam procedur

3.2.1. main

Vstupní bod aplikace
Adresa: 0x0040011C

Provede volání procedur, které zapříčiní zeptání se uživatele na vstup. Na základě vstupu vypíše požadovaný počet členů fibonacciho posloupnosti.

3.2.2. crlf

Vypisuje řídící znak konce řádku
Adresa: 0x00400024

Vypisuje znak konce řádku. Ten je uložen v paměti jako text, který začíná na adrese na kterou ukazuje návěští nl. Text je vypsán systémovým voláním služby 4 (print_str), která jako parametr přijímá adresu počátku řetězce.

3.2.3. fibbonaci

Vypíše požadovaný počet členů posloupnosti
Adresa: 0x00400054

Nejprve se za pomocí systémového volání (4 = print_str) vypíše text, kterým po uživateli vyžadujeme zadání počtu členů. Tento vstup se následně ověří kontrolou zda bylo číslo zadáno v požadovaném intervalu 2-48. V případě, že je vstup mimo interval, program automaticky nastaví vstup na nejbližší použitelný počet.

Výpočet a výpis jsou prováděny uvnitř cyklu. Nejprve se uloží registr, ve kterém je uložena hodnota vyššího členu posloupnosti do registru pomocného, následně se do registru vyššího členu přičte obsah registru, který reprezentuje předchozí člen posloupnosti. Následně přesuneme obsah pomocného registru do registru prvního členu. systémového volání (1 = print_int) vypisujeme vždy obsah registru vyššího členu po jeho výpočtu. Celý výše uvedený algoritmus opakujeme (x-2) krát. Odečítáme dva průchody vzhledem k výpisu prvních dvou členů mimo cyklus.

3.2.4. vypis

Vypíše text, obsahující otázku pro uživateli kolik členů chce vypsat
Adresa: 0x0040003C

Výpis je provádět systémovým voláním (4 = print_str). Jako parametr je předána adresa řetězce začínajícího na návěští msg.

4. Použité algoritmy

4.1. Výpočet členu posloupnosti

Při návrhu programu na výpočet a výpis fibonacciho posloupnosti byl použit následující Java algoritmus:

```
int a = 0;
int b = 1;
int c = 0;

System.out.println(a);
System.out.println(b);

Scanner in = new Scanner(System.in);

int num = in.nextInt();

for (int i = 1; i <= num - 2; i++) {
        c = b
        b = a + b;
        a = c;

        System.out.println(b);
}
```

Řešení v MIPS nevyužívá zásobníku, narozdíl od H8S zde máme dostatečné množství použitelných registrů. Nejprve se za pomocí systémového volání vypíše text, kterým po uživateli vyžadujeme zadání počtu členů. Tento vstup se následně ověří kontrolou zda bylo číslo zadáno v požadovaném intervalu 2-48. V případě, že je vstup mimo interval, program automaticky nastaví vstup na nejbližší použitelný počet.

Výpočet a výpis jsou prováděny uvnitř cyklu. Nejprve se uloží registr, ve kterém je uložena hodnota vyššího členu posloupnosti do registru pomocného, následně se do registru vyššího členu přičte obsah registru, který reprezentuje předchozí člen posloupnosti. Následně přesuneme

obsah pomocného registru do registru prvního členu. Za pomocí systémového přerušení vypisujeme vždy obsah registru vyššího členu po jeho výpočtu. Celý výše uvedený algoritmus opakujeme (x-2) krát. Odečítáme dva průchody vzhledem k výpisu prvních dvou členů mimo cyklus.

4.2. Převod dekadického čísla na ASCII

Naštěstí, MIPS obsahuje systémové volání, které nám umožní přímý výpis číselné hodnoty za pomocí systémového volání. Pro úplnost bych zde však rád uvedl kroky, které by lze bylo nutné provést v případě, že bychom použili jiný typ ASM pro jiný mikroprocesor, který tato systémová volání nepodporuje.

V jiných ASM, po té, co jsme spočítali člen k vypisu, budeme k němu muset vytvořit ASCII vyjádření.

Vlastně jde o to, že pokud máme v registru/paměti uloženo číslo 58350, nejedná se o jeho textovou podobu, tak je v tomto případě dekadicky 53 56 51 53 48. Ano z jednoho uloženého registru najednou musíme vytvářet více hodnot.

Tento problém jsem nakonec vyřešil tak, že jsem využil vlastnosti číselních soustav. Konkrétně dekadická soustava má pouze deset číslic 0 až 9. Čísla větší než 9 jsou tak složena z jednotlivých číslic. Počet číslic pak určuje nejvyšší řád čísla.

Zjednodušeně dokážeme číslo 58350 rozepsat na: $(5 * 10^4) + (8 * 10^3) + (3 * 10^2) + (5 * 10^1) + (0 * 10^0)$. Před výpisem vlastně chceme zjistit násobek základu 10^x

Toho docílíme tak, že budeme postupně celočíselně dělit daným základem 10^X. Výsledek dělení použijeme jako hodnotu číslice, kterou budeme chtít vypsat. Vzhledem k tomu, že v ASCII tabulce na hodnotách

0 až 9 nejsou znaky pro čísla, musíme k vypočtené hodnotě přičíst hodnotu 48. Tím dostaneme tisknutelnou číslici.

4.2.1. Převod čísla 58350 na ASCII - příklad

$58350 / 10^4 = 5$ (zbytek 8350)
vypíšeme číslici 5 (ASCII: 53)

$8350 / 10^3 = 8$ (zbytek 350)
vypíšeme číslici 8 (ASCII: 56)

$350 / 10^2 = 3$ (zbytek 50)
vypíšeme číslici 3 (ASCII: 51)
$50 / 10 = 5$ (zbytek 0)
vypíšeme číslici 5 (ASCII: 53)

$0 / 10^0 = 0$ (zbytek 0)
vypíšeme číslici 0 (ASCII: 48)

Algoritmus tedy převede číslo 58350 na sekvenci bajtů 53 56 51 53 48.

5. Obsluha

Program je ovládán přes textovou konzoli. Uživatel je vyzván k zadání počtu členů. V případě, že je vstup mimo přípustné hodnoty (určené intervalem <2,48>), upraví vstup na nejbližší hodnotu v intervalu - standardně supremum a infimum daného intervalu. O této změně je uživatel informován výstupem textu na konzoli.

6. Závěr

Program plně odpovídá zadání a je plně funkční. V programu byly využity hlavně registry, hodnoty se neukládají do paměti a nemusí se z ní při další iteraci cyklu znovu načítat, díky čemuž bude program o něco rychlejší než v případě implementace stejného zadání na procesoru H8S. Stále však musíme načítat uložený text z paměti.