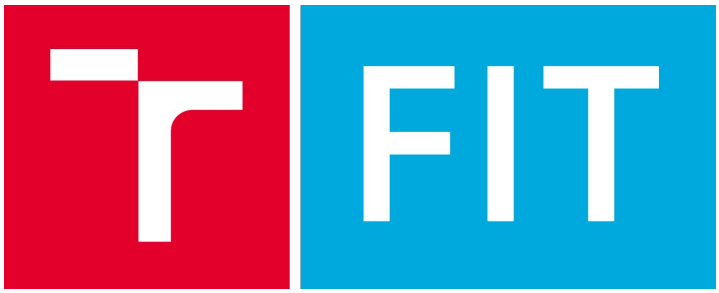
VYSOKÉ UČENIE TECHNICKÉ V BRNE

Fakulta informačných technológií



Mikroprocesorové a vstavané systémy

2015/2016

Simulácia v CodeWarrior

Svetelné noviny

Filip Gulán (xgulan00) 28.10.2015

Obsah

1 Úvod

Úlohou tohoto projektu bolo v jazyku C, alebo Assembler napísať pre True-Time Simulátor prostredí CodeWarrior verzia 6.x aplikáciu Svetelné noviny, určenú pre zobrazovanie textu na display zložený z LED usporiadaných do matice o 8 riadkoch a 32 stĺpcoch. Mali sme predpokladať implementáciu na platforme HC(S)08.

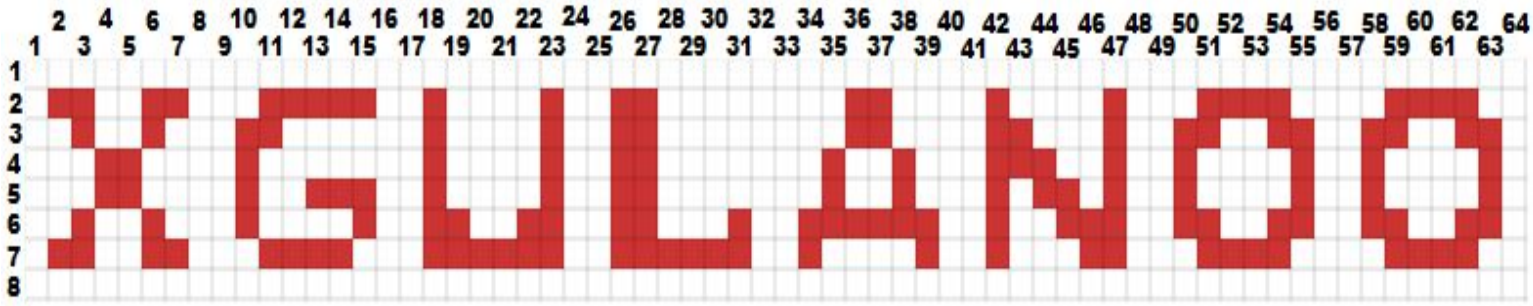
Aplikácia má byť schopná zobraziť na display časť môjho prihlasovacieho mena (xgulan00), uloženého v pamäti mikrokontroléru. Každý zo znakov loginu má reprezentovať bitmapu v rastre o veľkosti 8 riadkov a 8 stĺpcov.

2 Návrh a popis aplikácie

Nasledujúce riadky popisujú vlastné riešenie problematiky tohoto projektu. Popisujem tu návrh aplikácie a samotný algoritmus.

2.1 Návrh aplikácie

Program po spustení na začiatku zobrazuje prvé 4 písmená môjho loginu, teda xgul. Ostatné písmená sa nachádzajú taktiež v pamäti, avšak vo Visualisation tool ich nezobrazujeme. Jednotlivé stĺpce na obrázku č. 1 predstavujú adresu, na ktorej je nejaká hodnota. Stĺpcu 1 prislúcha adresa 612, stĺpcu 2 zase adresa 613… Riadky zase predstavujú jednotlivé bity hodnôt. Teda riadku, ktorý je označený ako jedna prislúcha 1., najmenej významný bit. Teda ak chceme rozsvietiť LEDku, v 2 riadku v 2 stĺpci, tak na adresu 513 vložíme hodnotu 00000010 (binárne, alebo číslo 2 decimálne). Stĺpce od 32 vyššie slúžia ako pomocné, pre ľahkú realizáciu rotácie.



Obrázok 1 - grafické znázornenie pamäte na začiatku

2.2 Popis aplikácie

Pri spustení programu sa najprv spustí funkcia *nastavAdresu()*, kde sa jednotlivým ukazateľom do pamäte nastaví miesto, kam ukazujú. Začiatok je od adresy 608 (v desiatkovej

sústave). Ukazateľe, ktoré ukazújú do miesta, kde sú namapované LEDky, sú uložené pre ľahšiu manipuláciu v poli. Po nastavení ukazateľov sa spustí funkcia *inicializuj()*, ktorá inicializuje pamäťové miesta na určité hodnoty (viz. obrázok 1). Ďalej nasleduje nekonečný for cyklus, v ktorom vykonávam akciu podľa vybraného módu. Mód sa vyberá vo funkcií *zistiMod()*, a následne sa mód nastaví pomocou funkcie *nastavMod()*. Podľa daného módu sa potom spúšťa buď funkcia *rotaciaDolava()*, *rotaciaDole() alebo inicializuj()*. Funkcia *rotaciaDolava()*, slúži na rotovanie loginu smerom doľava. Funkcia *rotaciaDole()*, slúži na rotovanie loginu smerom dole. O rýchlosť rotácie loginu sa stará funkcia *spomalenie()*.

3 Významné úseky kódov

Táto časť popisuje významné funkcie, ako rotaciaDolava() a rotaciaDole(). Ostatné funkcie programu sú dostatočne okomentované v zdrojovom kóde.

3.1 Funkcia *rotaciaDolava()*

Táto funkcia zabezpečuje rotáciu loginu doľava. Táto funkčnosť je zabezpečená tak, že hodnoty na adresách, ktoré reprezentujú stĺpec lediek sa poposúvajú smerom dole, teda hodnota na adrese 614 sa dá na adresu 613, hodnota na adrese 615 sa dá na adresu 614...

```
/*
* Funkcia, ktora zabezpecuje rotaciu loginu dolava
*/
void rotaciaDolava() {
 unsigned char pomocny = *login[0];
 int i;
 for(i = 0; i < 63; i++) {
   *login[i] = *login[i + 1];
 }
 *login[63] = pomocny;
}
```

3.2 Funkcia *rotaciaDole()*

Táto funkcia zabezpečuje rotáciu loginu dole. Funkcionalita je implementovaná tak, že sa skontroluje hodnota na každej adrese, ktorá reprezentuje stĺpec lediek, či je číslo väčšie ako 127, teda či sa na najviac významnom bite nachádza jednotka. Ak je číslo väčšie ako 127, tak hodnotu na danej adrese bitovo posunieme doľava a k tomuto číslu pričítame

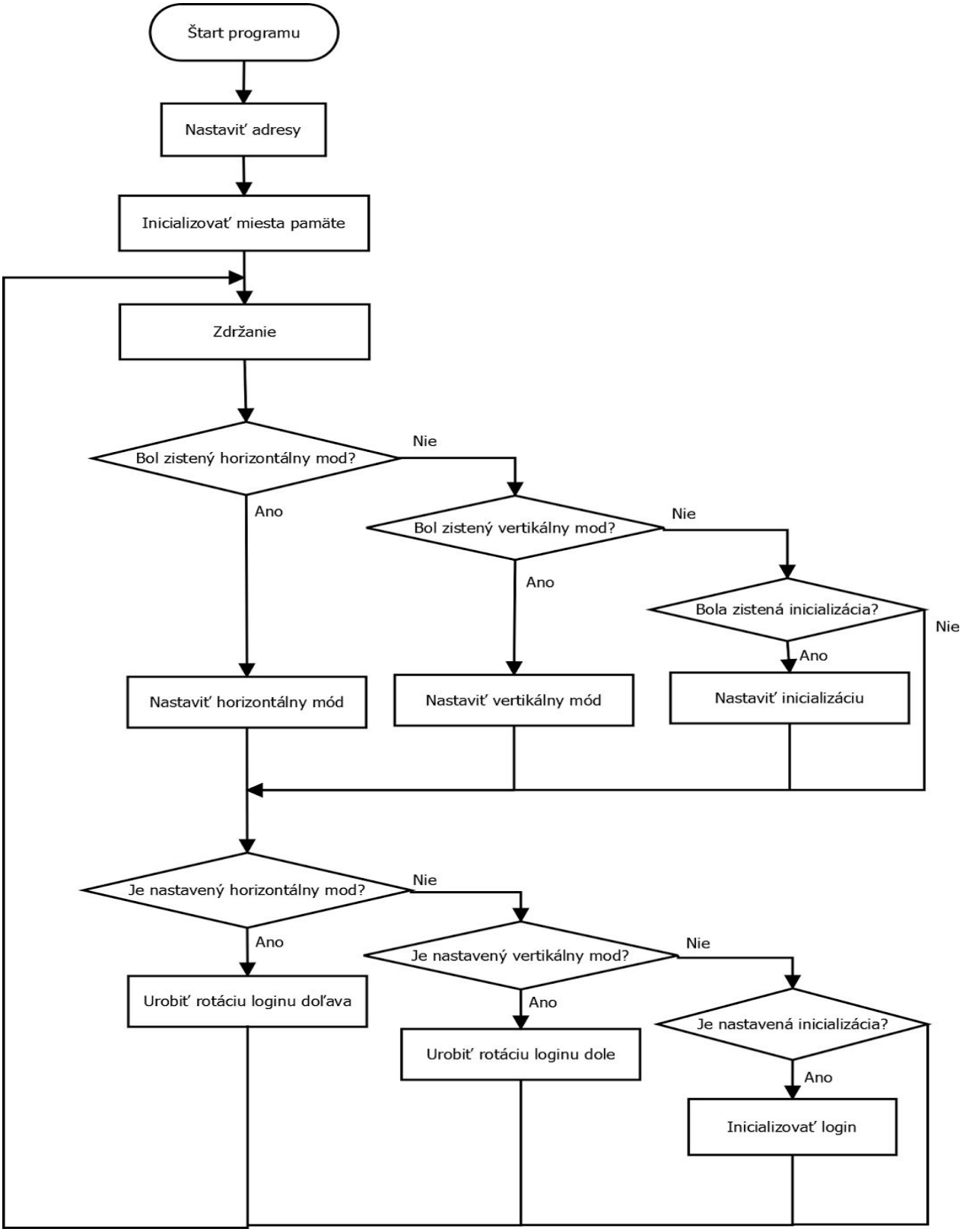
jednotku. Ak je číslo rovné, alebo menšie ako 127, tak hodnotu na adrese iba bitovo posunieme.

```
/*
* Funkcia zabezpecuje rotaciu loginu dole
*/
void rotaciaDole() {
 int i;
 for(i = 0; i < 63; i++) {
   if(*login[i] > 127) { //bitovy rotacia je tu urobena ako bitovy posun, ak je najvyssi bit 1, tak iba po posune pricitam 1ku
    *login[i] = *login[i] << 1;
    *login[i] = *login[i] + 1;
   } else {
     *login[i] = *login[i] << 1;
   }
 }
}
```

4 Analýza pamäťových nárokov

Nároky na RAM	680 B
Nároky na Flash pamäť	140 B
Veľkosť kódu	211 riadkov

5 Vývojový diagram



5 Záver

Program bol riadne otestovaný, behom tohoto testovania sa nevyskytli žiadne chyby a všetky mnou navrhnuté testy skončili úspešne. Program bol tvorený v prostredí CodeWarrior od firmy Freescale na školských počítačoch v CVT za pomoci školou vytvoreného a mnou upraveného Visualisation Tool.