PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Simulátor mikroprocesoru Microchip

2013 Michal Vojáček

Anotace

Cílem práce bylo vytvořit aplikaci simulující chování mikroprocesoru Microchip. Hlavní důraz byl kladen na co možná nejjednodušší ovládání a názornost, která chybí standardnímu vývojovému prostředí od výrobce mikroprocesoru.

Děkuji vedoucímu práce RNDr. Arnoštu Večerkovi za odborné vedení, cenné rady.

Obsah

Úvod **6**

1. Mikroprocesory Microchip **8**
1.1. Stručný popis mikroprocesoru PIC16F877 8
1.2. Instrukční sada 9

2. Uživatelská dokumentace **10**
2.1. Systémové požadavky 10
2.2. Instalace a spuštění aplikace 10
2.3. Odinstalování aplikace 10
2.4. Popis aplikace 10
2.4.1. Nabídkový pruh 10
2.4.2. Panel nástrojů 11
2.4.3. Program memory 11
2.4.4. Záložka Registr 11
2.4.5. Záložka Hardware 12
2.4.6. Stavový pruh 14
2.5. Použití aplikace 14

3. Programátorská dokumentace **15**
3.1. Použitá technologie 15
3.2. Architektura aplikace 15
3.2.1. Projekt mcuPIC16F877A 15
3.2.2. Projekt SimulatorPIC 19

Závěr **26**

Reference **27**

A. Popis instrukční sady procesorů PIC **28**
A.1. Instrukce nulování a nastavení 31
A.2. Instrukce přesunu dat 31
A.3. Instrukce podprogramů a přerušení 33
A.4. Instrukce skoků v programu 33
A.5. Zvláštní instrukce 34

B. Obsah přiloženého CD **36**

Seznam obrázků

1. Vnitřní architektura mikroprocesoru 8
2. Panel nástrojů 11
3. Hlavní okno aplikace - záložka Registr 12
4. Hlavní okno aplikace - záložka Hardware 13
5. Průběh kompilace 22
6. Vývojový diagram první fáze kompilace 24
7. Vývojový diagram druhé fáze kompilace 25

Úvod

Mikroprocesory od firmy Microchip Technology Inc.[1] označované zkratkou PIC (Peripheral Interface Controller) jsou programovatelné elektronické součástky, které jsou zaměřené především na realizaci měřících a řídících akčních členů řídících systémů. Programy pro tyto mikroprocesory jsou ve většině případů vytvářeny v aplikaci „MPLAB® X Integrated Development Environment (IDE)“ zdarma poskytované výrobcem mikroprocesorů.

Jedná se o velmi propracovaný vývojový nástroj podporující desítky tisíc typů 8,16,32-bitových mikroprocesorů, DSP mikroprocesorů, pamětí, převodníků a dalších komponent. Obsahuje kompilátor assembleru i jazyka C, logický analyzátor, osciloskop, generátory průběhů. Podporuje celou řadu simulátorů, debuggerů i hardwarové real-time emulátory. Tento obrovský arzenál funkcí je, ale dle mého názoru současně i jeho nevýhodou. Z vlastní zkušenosti totiž vím, jak těžké jsou začátky programování mikroprocesorů v takto komplexním nástroji. Začínající programátor mikroprocesorů potřebuje mnohem jednodušší nástroj pro psaní a především simulaci napsaných programů.

Začínající programátor, který se učí programovat v jakémkoliv programovacím jazyce začíná programem „Hello word“. Takové „Hello word“ ve světě mikroprocesorů Microchip, ARM, AVR, 8051, FPGA je rozblikání led diody na jednom z výstupních pinů. Simulace tohoto krátkého programu v nástroji „MPLAB® X IDE“ vyžaduje vytvoření celého projektu, včetně nastavení kompilátoru, zobrazovaných registrů, simulátoru, oscilátoru a dalších. To může být pro začínajícího programátora velkou překážkou. Proto jsem se při návrhu vlastního řešení zaměřil především na co možná nejjednodušší použití simulátoru, bez zbytečného nastavování spousty parametrů. Tedy použití ve stylu nahrej program a simuluj. Simulovaný hardware a jeho připojení jsem zvolil tak, aby v případě hardwarové realizace bylo možné využít běžně konstruované testovací desky jako jsou například „ChiponII“, „Olimex-proto boards“[2] a další.

Při výběru typu mikroprocesoru jsem se rozhodl pro osmibitový typ, protože je to nejčastější volba začínajících programátorů. I po tomto zúžení však stále zůstávalo na výběr několik set různých typů, lišících se počtem pinů, integrovaných periferií, velikostí a typem pamětí. Postupným zužováním výběru zůstaly dva typy a to PIC16F84 a PIC16F877.

Procesor PIC16F84 je historicky nejpoužívanější začínajícími programátory, ale jedná se už o velmi zastaralý typ s pouze 13 využitelnými piny bez jakýchkoliv integrovaných funkcí, proto jsem raději zvolil typ PIC16F877, který má až 35 využitelných pinů a spoustu integrovaných periferií, jako jsou například analogové vstupy, sériový port, PWM výstupy, komparátory a jiné. Zadání bakalářské práce

[1] `www.microchip.com`

[2] `www.olimex.com/Products/PIC/Proto/`

Zadání bakalářské práce

Cílem práce je sestavit simulátor pro některého typického zástupce řady jednočipových mikroprocesorů řady Microchip. Simulátor by měl umožňovat vykonávání vloženého programu po krocích a zobrazovat přitom údaje:

- vykonávanou instrukci a instrukce v jejím okolí
- obsah obecných registru
- obsah speciálních registru
- obsah paměti RAM
- úroveň signálu na portech K portům je možné připojit zobrazovací jednotku (řadu LED diod nebo segmentové zobrazovače).

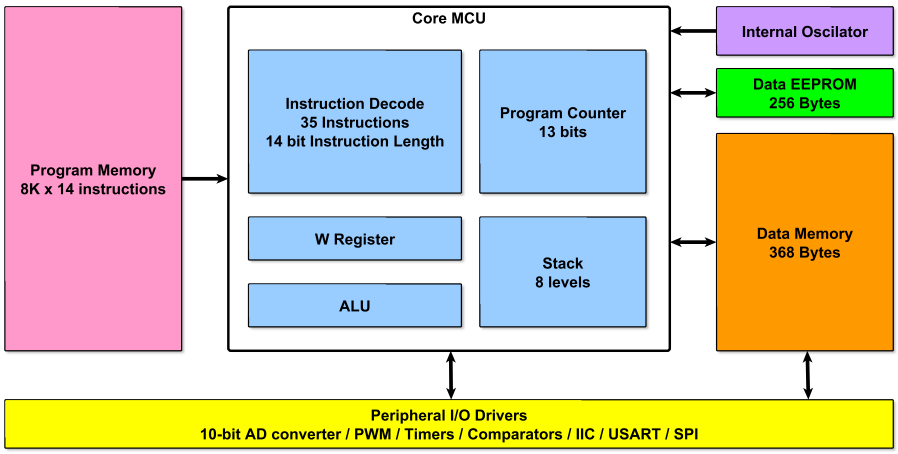
1. Mikroprocesory Microchip

Tvoří jednočipové mikropočítače tvořené jediným integrovaným obvodem, který v sobě obsahuje veškeré potřebné součástky pro samostatnou funkci. Firma Microchip Technology Inc. vyrábí mikroprocesory v několika základních řadách. Jsou to:

- PIC10, PIC12, PIC16 a PIC18 -osmibitové mikroprocesory.
- PIC24 -šestnáctibitové mikroprocesory.
- dsPIC16 -šestnáctibitové digitální signálové mikroprocesory (DSP).
- PIC32 -tvoří nejvýkonnější řadu dvaatřicetibitových mikroprocesorů.

1.1. Stručný popis mikroprocesoru PIC16F877

Mikroprocesor PIC16F877 je osmibitový mikroprocesor postavený na Harvardské architektuře, to znamená, že paměť programu a dat je vzájemně oddělená a má různou šířku datového slova. Jádro mikroprocesoru využívá redukovanou sadu instrukcí pevné délky (14-bit) tak zvaný RISC. Kromě instrukcí podmíněných skoků vykonává všechny strojové instrukce v jediném cyklu (jeden cyklus = čtyři hodinové takty oscilátoru). Obsahuje jediný pracovní registr W (Work), přes který jsou prováděny veškeré aritmetické a logické operace. Osmiúrovňový hardwarový zásobník volání podprogramu. Veškeré konfigurační registry jsou mapovány do paměti RAM, proto se s nimi pracuje stejně jako s datovou pamětí. Dále tento mikroprocesor obsahuje řadu hardwarových periferií jako jsou 10-bitový analogově digitální převodník, komparátory, časovače, EEPROM, sekundární oscilátor, PWM generátory, sériový a paralelní port, IIC sběrnice a jiné.1.



Obrázek 1. Vnitřní architektura mikroprocesoru

1.2. Instrukční sada

Instrukční sada osmibitových mikroprocesorů obsahuje 35 strojových instrukcí. Celý instrukční soubor je symetrický, to znamená, že lze provádět jakoukoliv instrukci na jakémkoliv registru. Instrukce jsou rozděleny do těchto šesti skupin:

- Aritmetické a logické operace
- Instrukce nulování a nastavení
- Instrukce přesunu dat
- Instrukce skoků v programu
- Instrukce podprogramů a přerušení
- Zvláštní instrukce

Podrobný popis instrukcí mikroprocesoru je uveden v příloze A.

2. Uživatelská dokumentace

2.1. Systémové požadavky

Aplikace je určena pro operační systémy Microsoft verze Windows Vista a Windows 7 s minimálním zobrazovacím rozlišením 1280 x 800 pixelů. Pro běh aplikace je také nutná instalace rozhraní .NET Framework 4.5.

2.2. Instalace a spuštění aplikace

Aplikaci nainstalujete z CD-ROM disku spuštěním instalačního souboru „setup.exe“ umístěném v adresáři „INSTALL“. Dále postupujte dle pokynů instalátoru.

V případě absence rozhraní .NET Framework 4.5, se instalátor pokusí toto rozhraní stáhnout z webových stránek společnosti Microsoft a nainstalovat. Instalátor na pracovní ploše vytvoří zástupce pro spuštění aplikace se jménem „SimulátorPIC“. Stejnojmenný zástupce bude také vytvořen v nabídce „Start“ pod volbou „Všechny programy“ → „SimulátorPIC“.

2.3. Odinstalování aplikace

Aplikaci odinstalujete standardním způsobem a to tak, že v nabídce „Start“ →„Ovládací panely“ → „Programy a funkce“ vyberete položku „SimulátorPIC“ a kliknete na tlačítko „Odinstalovat nebo změnit“. V otevřeném okně vyberte volbu „Odebrat aplikaci z tohoto počítače“ a potvrďte tlačítkem „OK“.

2.4. Popis aplikace

Hlavní okno aplikace se skládá z těchto částí: Nabídkový pruh, Panel nástrojů, Výpis obsahu programové paměti, Záložka Registr, Záložka Hardware.

2.4.1. Nabídkový pruh

- File

  Open - slouží k nahrávání simulovaného programu v assembleru (textový soubor s příponou *.asm).

  Exit - ukončení aplikace.

- Help

  Help - otevře okno s popisem aplikace.

  Website www.microchip.com - otevře webové stránky výrobce mikroprocesoru.

Datasheet PIC16F87xA - otevře dokumentaci simulovaného mikroprocesoru.

About - otevře okno s informacemi o aplikaci.

2.4.2. Panel nástrojů

Obrázek 2. Panel nástrojů

Start/Stop - tlačítko spustí nebo zastaví simulaci.

Step - stisknutí tlačítka provede jednu instrukci.

Reset - tlačítko vynuluje celý mikroprocesor i jeho periferie.

Simulator in STOP/RUN - informace o aktuální stavu simulátoru.

Simulation speed - nastavení rychlosti simulace.

Basic clock MCU - informace o kmitočtu použitého oscilátoru.

Instruction time - čas trvání jedné instrukce.

Total cycles - počet provedených instrukcí od počátku simulace.

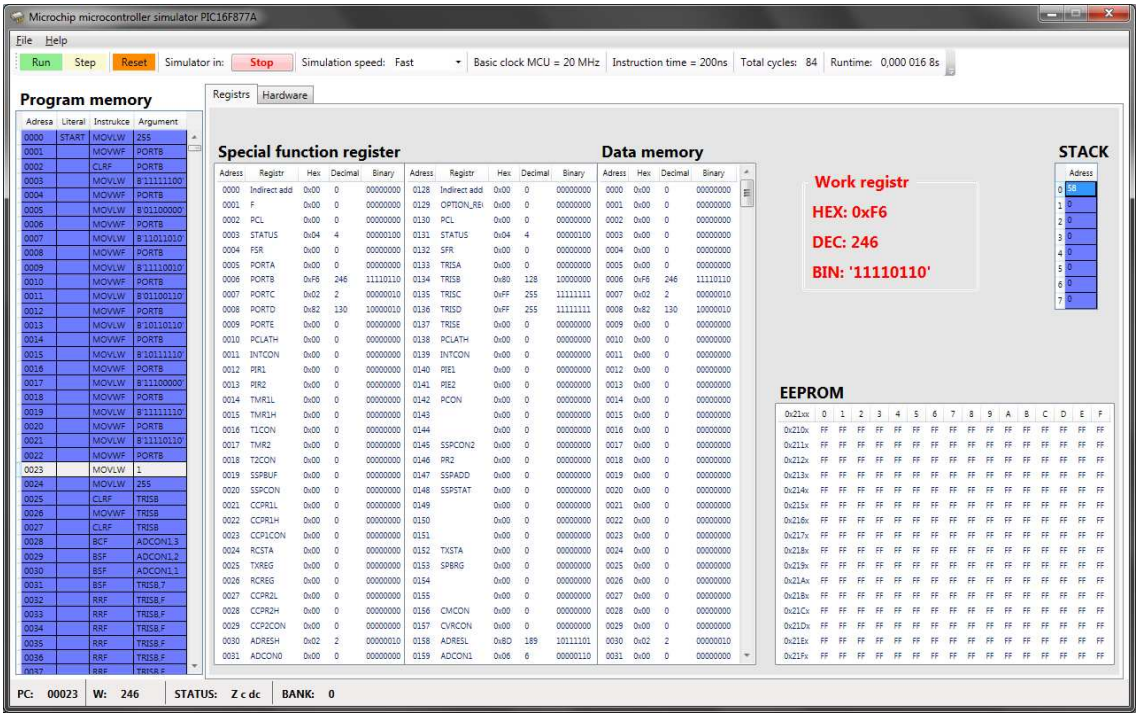
Run time - skutečný čas běhu od počátku simulace.

2.4.3. Program memory

Tato oblast zobrazuje celou programovou paměť. Každý řádek zobrazuje jedno adresovatelné slovo v programové paměti. V prvním sloupci je desítková adresa. Druhý sloupec slouží pro zobrazení pojmenovaného návěstí. Ve třetím sloupci je uvedena instrukce a poslední sloupec obsahuje data.

2.4.4. Záložka Registr

Speciální funkční registry (SFR) - v prvních dvou částech jsou zobrazeny speciální funkční registry (ty jsou adresovány přímo do datové paměti). Všechny hodnoty jsou zobrazeny s příslušnou desítkovou adresou a hodnota je zobrazena v šestnáctkové, desítkové i binární podobě.



Obrázek 3. Hlavní okno aplikace - záložka Registr

Datová paměť - ve třetí části je zobrazena celá adresovatelná část datové paměti RAM (pro úplnost včetně částí zobrazených v SFR). Zde jsou také všechny hodnoty zobrazeny s příslušnou desítkovou adresou a hodnota je zobrazena v šestnáctkové, desítkové i binární podobě.

Stack - zobrazuje obsah osmiúrovňového zásobníku a ukazatel na jeho vrchol.

EEPROM - zobrazuje obsah paměti EEPROM v šestnáctkové soustavě.

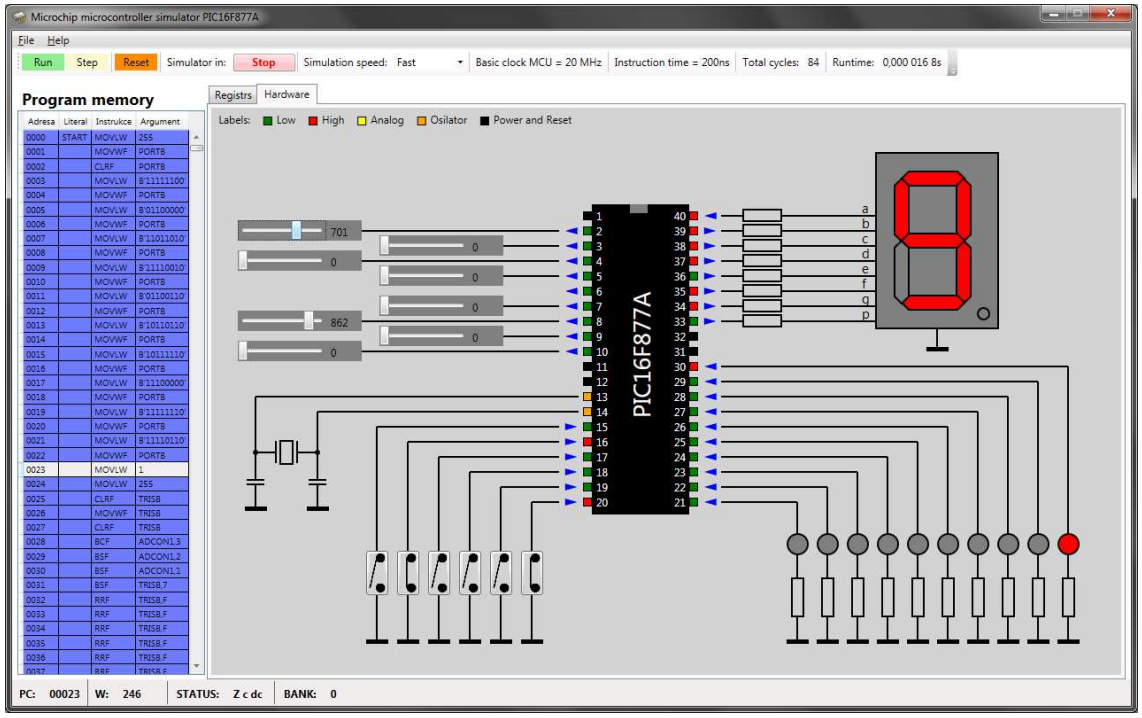
Work regisrt - ukazuje aktuální hodnotu uloženou v pracovním registru procesoru v šestnáctkové, desítkové i binární podobě.

2.4.5. Záložka Hardware

Mikroprocesor - je zobrazen v pouzdru 40-Pin PDIP při pohledu shora. Samotné piny mikroprocesoru jsou barevně rozlišeny a to následovně:

Zelený pin - označuje vstupní nebo výstupní digitální pin s nízkým potenciálem (logická 0).

Červený pin - označuje vstupní nebo výstupní digitální pin s vysokým potenciálem (logická 1).



Obrázek 4. Hlavní okno aplikace - záložka Hardware

Žlutý pin - označuje vstup použitý jako vstupní kanál pro desetibitový analogově digitální převodník.

Oranžové piny - slouží pro připojení externího oscilátoru (v tomto případě krystalový oscilátor pracující na kmitočtu 20 MHz)

Černý pin - značí napájecí nebo resetovací piny (nejsou podstatné pro simulaci).

V těsné blízkosti pinů jsou zobrazeny modré šipky, které ukazují zda je pin nastaven jako vstupní (šipka ukazující směrem do mikroprocesoru) nebo výstupní (šipka ukazující směrem ven z mikroprocesoru).

Posuvníky - symbolizují tahové potenciometry připojené na piny, které lze nakonfigurovat jako vstupy pro desetibitový analogově digitální převodník.

Krystalový oscilátor s kondenzátory -je zobrazen pouze pro úplnost schématu symbolizujícího zapojení mikroprocesoru.

Spínače - používané pro interakci se simulovaným programem.

LED diody signalizují stav výstupních pinů mikroprocesoru.

Sedmisegmentový displej používaný pro zobrazení čísel a dalších znaků.

2.4.6. Stavový pruh

Stavový pruh zobrazuje:

Hodnotu programového čítače

Obsah pracovního registru W

Příznaky stavového registru STATUS - dle prostředí MPLAB IDE jsou neaktivní příznaky symbolizovány malými znaky a aktivní velkými znaky (Z-zero, C-carry, DC-digit carry)

Aktuálně používanou paměťovou banku

2.5. Použití aplikace

Celá aplikace byla navržena tak, aby její použití bylo co možná nejjednodušší. Proto je potřeba udělat jen minimální množství kroků pro zahájení simulace programu.

Po spuštění programu stačí v záložce „File“ → „Open…“ otevřít textový soubor s kódem v assembleru určeným pro mikroprocesor PIC16F877A s koncovkou „*.asm“. Pro první seznámení s aplikací je po instalaci připraven testovací program „test.asm“.

Po otevření se k tomuto souboru automaticky připojí soubor „pic16f877a.inc“, to je inicializační soubor přímo od firmy Microchip Technology Inc., ten mimo jiné zavede do kompilátoru symbolické názvy a odpovídající adresy SFR registrů. Takže ve vlastním zdrojovém kódu již není potřeba znovu a znovu definovat desítky nutných parametrů.

V případě, že se během kompilace v kódu vyskytne syntaktická chyba dojde k přerušení kompilace a program nahlásí řádek na kterém k chybě došlo. Dokud není bezchybně zkompilován kód nelze přistoupit k simulaci.

Po bezchybné kompilaci se zkompilovaný kód zapíše do programové paměti mikroprocesoru. Nyní stačí kliknout na tlačítko „RUN/STOP“ a pozorovat průběh simulace.

3. Programátorská dokumentace

3.1. Použitá technologie

Aplikace je napsána v programovacím jazyce C# využívající technologii Microsoft.NET Framework 4.5 a pro grafické rozhraní je použita technologie Windows Presentation Foundation (WPF)

Z použité technologie Microsoft.NET Framework 4.5 vyplývá potřebný operační systém od firmy Microsoft ve verzi Windows Vista, Windows 7 nebo Window 8. Aplikace byla vytvořena ve vývojovém prostředí Microsoft Visula Studio 2012.

3.2. Architektura aplikace

Celé řešení aplikace je umístěno do jmenného prostoru „SimulatorPIC“ a je rozděleno do dvou samostatných projektů. Projekt „mcuPIC16F877A“ obsahující celou logiku simulovaného procesoru a projektu „SimulatorPIC“ obsahující grafické uživatelské rozhraní aplikace.

3.2.1. Projekt mcuPIC16F877A

Projekt implementuje logiku simulovaného mikroprocesoru. Její architektura kopíruje skutečné vnitřní zapojení mikroprocesoru (viz obrázek 1.). Kde je každá vnitřní část mikroprocesoru reprezentována vlastní třídou.

Třída ALU - implementuje aritmeticko-logickou jednotku mikroprocesoru, obsahuje implementaci logických funkcí a skládá se z následujících částí:

enum BitOperation setBit, resetBit, invertBit výčet pro určení požadované operace.

setBit - nastaví bitu na logickou 1

resetBit - nastavení bitu na logickou 0

invertBit - inverze logické hodnoty bitu

byte BitSet(byte b, int bit) - vrací vstupní byte s požadovaným bitem nastaveným na logickou 1

byte BitReset(byte b, int bit) - vrací vstupní byte s požadovaným bitem nastaveným na logickou 0

bool IsBitSet(byte b, int bit) - testuje požadovaný bit ve vstupním bytu a vrací true pro bit nastavený na 1 a false pro bit nastavený na 0

void BitModifi(int adress, int bit, BitOperation operation)
- pro provádění bitových operací, jako své argumety vyžaduje určení O, kterou adresu a který bit se má modifikovat, poslední argument určuje prováděnou operaci

void StatusSet(bool zero, bool carry, bool digitCarry)
- nastaví bity příznakového registru STATUS dle vstupních argumentů pro Z, C a DC

void StatusSetZero(bool zero) - nastaví bit Z v příznakovém registru STATUS

void StatusSetCarry(bool carry) - nastaví bit C v příznakovém registru STATUS

Třída Stack - implementuje osmiúrovňový programový zásobní

int Pop() - metoda vrací adresu uloženou na vrcholu zásobníku

void Push(int adress) - metoda vkládá adresu na vrchol zásobníku

Třída ProgramMemory - implementuje programovou paměť mikroprocesoru

Struktura :

struct Word(...) - struktura obsahující potřebné informace pro zobrazení a provádění programu

lit - popisný název návěstí

instName - popisný název instrukce

pText - popisný název argumentů instrukce

i - identifikátor instrukce

p1 - první argument instrukce

p2 - druhý argument instrukce

Metody :

void Reset() - vyresetuje celou programovou paměť

void WriteToProgramMemory(Word word, int adress)
- zapíše word na příslušnou adresu v programové paměti

Word ReadFromProgramMemory(int adress)
- přečte word z příslušné adresy programové paměti

Třída DataMemory - implementuje datovou(RAM) paměť mikroprocesoru

Vlastnosti :

int LastModifyAdress - vrací adresu, na které proběhla poslední změna

Metody :

void Reset() - vyresetuje celou datovou paměť

byte ReadFromDataMemory(int adress) - vrací hodnotu uloženou na dané adrese datové paměti

void WriteToDataMemory(int adress, byte b) - zapíše byte na příslušnou adresu v datové paměti

void MirrorMemory(int adress, byte b) - v případě, že je modifinkovaná adresa zrcadlena i do jiných části paměti, zajistí zápis hodnoty i do nich

void SetMirror(int index, byte b) - metoda vypočítá zrcadlové adresy a zapíše na ně novou hodnotu

Třída DataEEPROM - implementuje EEPROM paměť mikroprocesoru, slouží pro záznam dat uložených i po výpadku napájení mikroprocesoru

Vlastnosti :

int LastModifyAdress - vrací adresu na které proběhla poslední změna

Metody :

void Reset() - vyresetuje celou EEPROM paměť

byte ReadFromDataEEPORM(int adress) - vrací hodnotu uloženou na dané adrese datové paměti

void WriteToDataEEPROM(byte b, int adress) - zapíše byte na příslušnou adresu v datové paměti

Statická třída Definitions - mimo níže popsané metody zavádí potřebné konstanty, jako jsou adresy speciálních funkčních registrů, zavedené značení, pozice speciálních bitů, seznamy instrukcí a jiné.

static int GetIdInstruction(string instructionName)
- pokud vstupní argument obsahuje jméno existující instrukce, vrátí její číselný identifikátor

Třída Core - implementuje jádro mikroprocesoru včetně samotných instrukcí, její konstruktor zavádí ostatní výše popsané části mikroprocesoru.

Vlastnosti :

byte W - work registr

byte PC - programový čítač

int BANK - paměťová banka

byte STATUS - příznakový registr

UInt64 TotalCycles - počet provedených instrukcí mikroprocesoru

Double ElapsedTime - čas běhu programu v reálném mikroprocesoru s kmitočtem primárního oscilátoru 20 MHz

Metody :

void Clock() - metoda zajistí provedení jedné instrukce mikroprocesoru

void ProcessingInstruction() - provede instrukci v programové paměti, na kterou ukazuje programový čítač (PC)

void ADConverter() - z použitého kanálu A/D převodníku přečte hodnotu a zapíše ji do speciálních registrů ADRESSL a ADRESSH

Metody implementující instrukce mikroprocesoru. - popis instrukcí viz příloha A.

void ADDLW(Word word)
void ADDWF(Word word)
void ANDLW(Word word)
void ANDWF(Word word)
void CONF(Word word)
void DECF(Word word)
void INCF(Word word)
void IORLW(Word word)
void IORWF(Word word)
void SUBLW(Word word)
void SUBWF(Word word)
void XORLW(Word word)
void XORWF(Word word)
void BCF(Word word)
void BSF(Word word)
void CLRF(Word word)
void CLRW(Word word)
void CLRWDT(Word word)
void MOVF(Word word)
void MOVLW(Word word)
void MOVWF(Word word)
void RLF(Word word)
void RRF(Word word)
void SWAPF(Word word)
void CALL(Word word)

void RETLW(Word word)
void RETURN(Word word)
void RETFIE(Word word)
void BTFSC(Word word)
void BTFSS(Word word)
void DECFSZ(Word word)
void GOTO(Word word)
void INCFSZ(Word word)
void NOP(Word word)
void SLEEP(Word word)

3.2.2. Projekt SimulatorPIC

Implementuje grafické rozhraní simulátoru mikroprocesoru a kompilátor. Využívá technologii Windows Presentation Foundation (WPF). Z toho vyplývá oddělení kódu pro definici vzhledu aplikace od kódu s funkční logikou. Pro definici vzhledu aplikace byl použit značkovací jazyk XAML. Kód funkční logiky je napsán v jazyce C#.

MainWindow.xaml a MainWindow.xaml.cs -implemetuje vzhled a interakční logiku aplikace.

void OneStepButton(object sender, RoutedEventArgs e)
- provede jednu instrukci

void StartStopButton(object sender, RoutedEventArgs e)
- zahájí nebo ukonči simulaci

void ResetButton(object sender, RoutedEventArgs e)
- provede reset mikrokontroleru a všech jeho periferií

void SimulationSpeed(object sender, SelectionChanged-

EventArgs e) - změna zvolené rychlosti simulace

void MenuItem_Open(object sender, RoutedEventArgs e)
- otevře dialogové okno pro načtení zdrojového souboru

void MenuItem_Close(object sender, RoutedEventArgs e)
- ukončí aplikaci

void MenuItem_Help(object sender, RoutedEventArgs e)
- otevře okno s nápovědou

void Open_Datasheet(object sender, RoutedEventArgs e)
- otevře datasheet mikroprocesoru PIC16F877A

void Open_Microchip(object sender, RoutedEventArgs e)
- otevře webové stránky výrobce mikroprocesoru

void AboutWindow_click(object sender, RoutedEventArgs e) - otevře okno s informacemi o aplikaci

void RefreshGUIAll() - provede kompletní obnovu všech částí GUI

void RefreshGUILastChange() - provede obnovení změněných hodnot GUI

void ShowProgramMemory() - zobrazí programovou paměť

void RefreshStatusBar() - obnoví zobrazované informace na statusBar

void ShowEEPROM() - zobrazí EEPROM paměť

void RefreshEEPROM() - obnoví EEPROM paměť

void ShowSFR() - zobrazí obsah speciálních funkčních registrů

void RefreshSFR() - obnoví zobrazované informace speciálních funkčních registrů

void ShowDataMemory() - zobrazí datovou paměť RAM

void RefresDataMemory() - obnoví zobrazované informace paměti RAM

string GetSubstitutionName(int adress) - metoda vrátí jméno návěstí - pokud je daná adresa pojmenovaná

void SwitchC0_click(object sender, RoutedEventArgs e) - metoda obsluhující změnu stavu přepínače připojeného k portu C0

void SwitchC1_click(object sender, RoutedEventArgs e) - metoda obsluhující změnu stavu přepínače připojeného k portu C1

void SwitchC2_click(object sender, RoutedEventArgs e) - metoda obsluhující změnu stavu přepínače připojeného k portu C2

void SwitchC3_click(object sender, RoutedEventArgs e) - metoda obsluhující změnu stavu přepínače připojeného k portu C3

void SwitchD0_click(object sender, RoutedEventArgs e) - metoda obsluhující změnu stavu přepínače připojeného k portu D0

void SwitchD1_click(object sender, RoutedEventArgs e) - metoda obsluhující změnu stavu přepínače připojeného k portu D1

void AnalogA0_change(object sender, RoutedPropertyChanged e) - metoda obsluhující změnu nastavení potenciometru připojené na port A0

void AnalogA1_change(object sender, RoutedPropertyChange e) - metoda obsluhující změnu nastavení potenciometru připojené na port A1

void AnalogA2_change(object sender, RoutedPropertyChanged e)
- metoda obsluhující změnu nastavení potenciometru připojené na port A2

void AnalogA3_change(object sender, RoutedPropertyChanged e)
- metoda obsluhující změnu nastavení potenciometru připojené na port A3

void AnalogA4_change(object sender, RoutedPropertyChanged e)
- metoda obsluhující změnu nastavení potenciometru připojené na port A4

void AnalogE0_change(object sender, RoutedPropertyChanged e)
- metoda obsluhující změnu nastavení potenciometru připojené na port E0

void AnalogE1_change(object sender, RoutedPropertyChanged e)
- metoda obsluhující změnu nastavení potenciometru připojené na port E1

void AnalogE2_change(object sender, RoutedPropertyChanged e)
- metoda obsluhující změnu nastavení potenciometru připojené na port E2

void SimulationDelay_DoWork(object sender, DoWorkEventArgs e)
- background worker vytvářející zvolenou časovou prodlevu mezi instrukcemi

void SimulationDelay_RunWorkerCompleted(object sender,

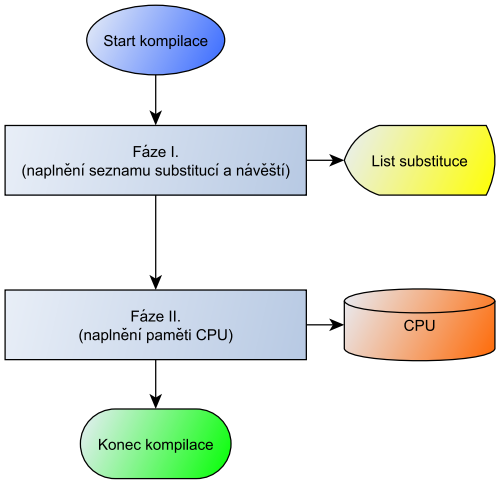
RunWorkerCompletedEventArgs e) - provede obnovení zobrazovaných hodnot a spuštění dalšího cyklu

Compiler.xaml a Compiler.xaml.cs - implementuje kompilátor programu. Kompilátor načítá zdrojové soubory a kontroluje správnost syntaxe. Zobrazuje detailní informace o průběhu kompilace.

Z důvodu kompatibility s originálním vývojovým prostředím MPLAB IDE je kompilátor vytvořen tak, aby zpracovával stejné zdrojové soubory, včetně řídících direktiv a všech zavedených způsobů zápisu binárních, desítkových a šestnáctkových čísel.

Kompilace probíhá ve třech krocích (viz obrázek 5.). V prvním kroku proběhne naplnění substituční tabulky (viz obrázek 6.) z inicializačního souboru mikroprocesoru od firmy Microchip. Ve druhém kroku dojde k doplnění substituční tabulky z kompilovaného souboru. A ve třetím kroku již dochází k nahrazování zavedených substitucí a plnění programové paměti (viz obrázek 7.).

void CompilePhaseI(string fileName, DoWorkEventArgs e)
- načte vstupní soubor a spustí první fázi kompilace



Obrázek 5. Průběh kompilace

void CompilePhaseII(string fileName, DoWorkEventArgs e)
- načte vstupní soubor a spustí druhou fázi kompilace

void CompileI(StreamReader configFile, DoWorkEventArgs e)
- první fáze kompilace (naplnění tabulky substitucí)

void CompileII(StreamReader configFile, DoWorkEventArgs e)
- druhá fáze kompilace (naplnění programové paměti)

string ReplaceSubstitutions(string line) - nahradí substituc ve vstupním řetězci

string MySplit(char separators, string line) - rozdělí řetězec dle vstupních znaků

bool IsNumber(string input) - vrací true pokud řetězec reprezentuje číslo v jakémkoliv formátu

int TryGetNumber(string input, int adress) - pokusí se vrátit číslo, pokud není vyvolá vyjímku new Exception("Error format number!");

bool IsNewSubstitution(string input) - vrací true pokud se jedná o novou substituci

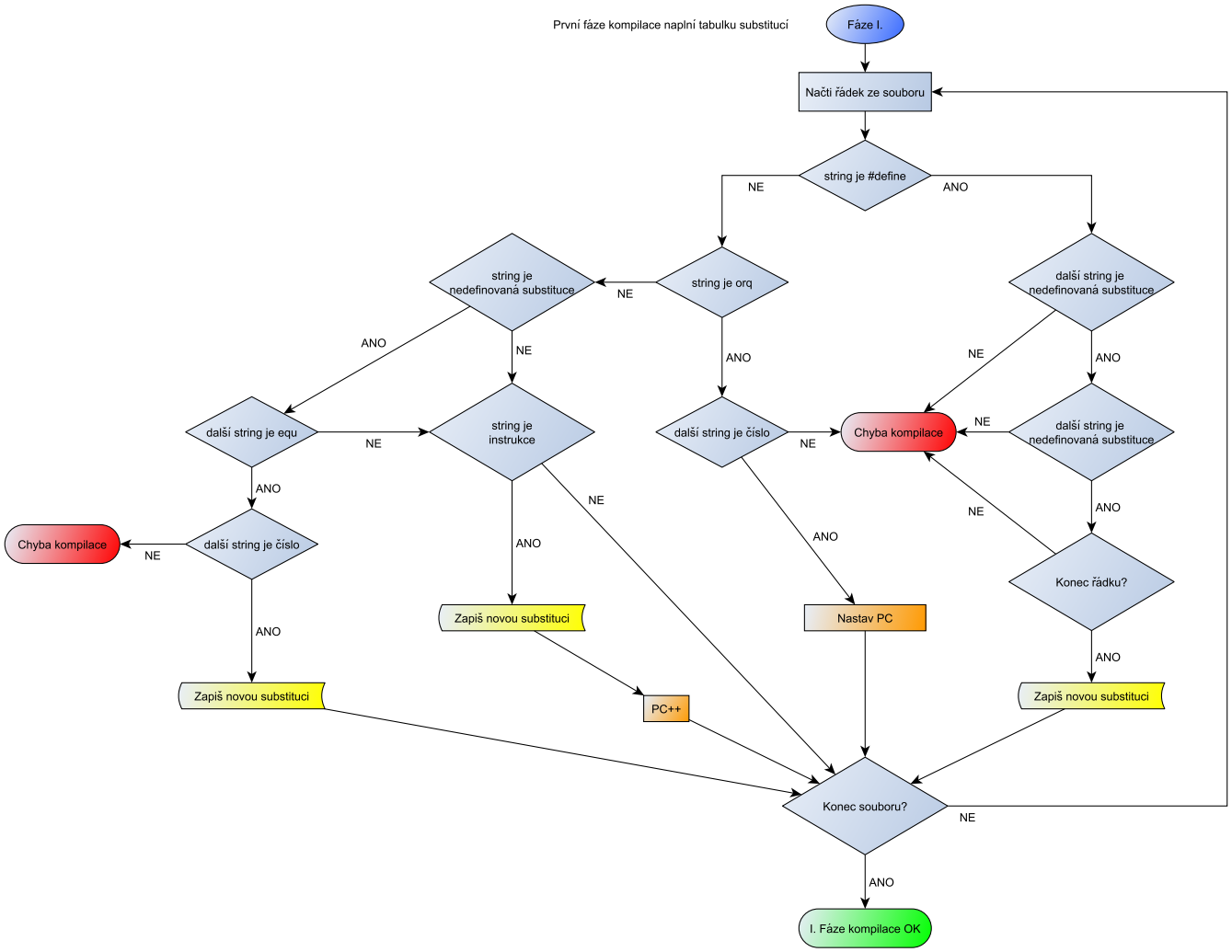
bool IsSubstitution(string input) - vrací true pokud se jedná o substituci

bool IsInstruction(string input) - vrací true pokud se jedná o instrukci

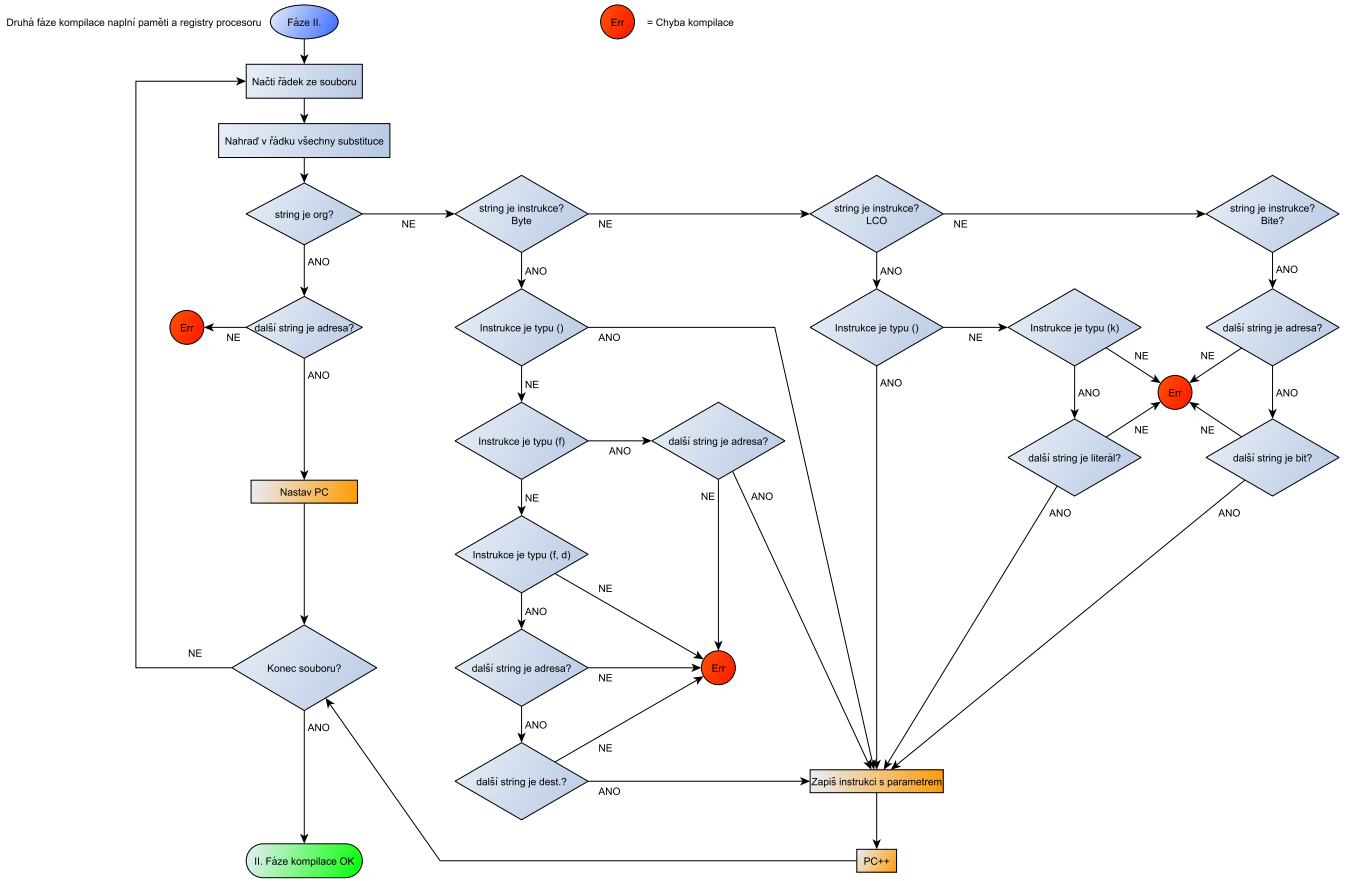
bool IsDirective(string input) - vrací true pokud se jedná o direktivu

HelpWindow.xaml a HelpWindow.xaml.cs - implementuje okno s nápovědou aplikace.

AboutWindow.xaml a AboutWindow.xaml.cs - implementuje okno s informacemi o aplikaci.



Obrázek 6. Vývojový diagram první fáze kompilace



Obrázek 7. Vývojový diagram druhé fáze kompilace

Závěr

Tato páce měla za úkol vytvořit jednoduchý a přehledný nástroj pro simulaci zdrojového kódu v assembeleru osmibitového mikroprocesoru firmy Microchip. Vytvořená aplikace "Simulátor PIC" provádí tyto funkce:

- načtení zdrojového kódu v assembleru
- syntaktickou kontrolu zdrojového kódu
- použití všech direktiv zavedených firmou mikroprocesoru
- zavádí inicializační soubor se základní konfigurací mikroprocesoru
- zobrazuje veškeré vnitřní části mikroprocesoru jako jsou programová paměť, datová paměť, EEPROM paměť, speciální funkční registry, zásobník atd.
- zobrazuje nastavení a aktuální stavy vstupně výstupních pinů
- implementuje v procesoru hardwarově integrovaný analogově digitální převodník
- poskytuje základní hardwarové prvky připojené k pinům mikroprocesoru jako jsou LED diody, přepínače, sedmisegmentový displej a potenciometry

V budoucnu by bylo možné rozšířit simulátor především o další v procesoru integrované periferie jako je například komparátor, PWM generátor, sériový a paralelní port.

Reference

[1] Hrbáček, Jiří. *Mikrořadiče PIC16CXX a vývojový kit PICSTART.* Ben, Praha, 1998.

[2] Vacek, Václav. *Učebnice programová PIC.* Ben, Praha, 2000.

[3] Peroutka, Oldřich. *Mikrokontroléry PIC16F87X.* Ben, Praha, 2005.

[4] `http://ww1.microchip.com/downloads/en/devicedoc/39582b.pdf` *PIC16F877XA Data sheet* Elektronická publikace, 2003.

[5] `www.microchip.com`

A. Popis instrukční sady procesorů PIC

V této kapitole jsou shrnuty instrukční sady procesoru PIC ...

Aritmetické a logické operace

ADDLW	ADD Literal and W
Zápis:	ADDLW k
Operace:	(W + k) →W
Popis:	Sečte obsah registru W s konstantou k, výsledek uloží do registru W.
Ovlivňuje:	C, DC, Z

ADDWF	ADD W and F
Zápis:	ADDWF f, d
Operace:	(W + f) → d
Popis:	Sečte obsah registrů W a f, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	C, DC, Z

ANDLW	AND Literal and W
Zápis:	ANDLW k
Operace:	(k & W) → W
Popis:	Provede logický součin registru W s konstantou k, výsledek uloží do registru W.
Ovlivňuje:	Z

ANDWF	AND W with F
Zápis:	ANDWF f, d
Operace:	(W & f) → f, d
Popis:	Provede logický součin obsahu registru f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

COMF	COMplement F
Zápis:	COMF f, d
Operace:	(f) → d
Popis:	Zamění jedničky a nuly v obsahu registru f (jedničkový doplněk čísla) a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

DECF	DECrement F
Zápis:	DECF f, d
Operace:	(f - 1) → d
Popis:	Odečte jedničku od obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

INCF	INCrement F
Zápis:	INCF f, d
Operace:	(f + 1) → d
Popis:	Přičte jedničku k obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

IORLW	Inclusive OR Literal with W
Zápis:	IORLW k
Operace:	(W or k) → W
Popis:	Provede logický součet (OR) obsahu registru W s konstantou k, výsledek uloží do registru W.
Ovlivňuje:	Z

INCF	INCrement F
Zápis:	INCF f, d
Operace:	(f + 1) → d
Popis:	Přičte jedničku k obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

IORLW	Inclusive OR Literal with W
Zápis:	IORLW k
Operace:	(W or k) → W
Popis:	Provede logický součet (OR) obsahu registru W s konstantou k, výsledek uloží do registru W.
Ovlivňuje:	Z

IORWF	Inclusive OR W with F
Zápis:	IORWF f, d
Operace:	(W or f) → f, d
Popis:	Provede logický součet (OR) obsahu registrů f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

IORWF	Inclusive OR W with F
Zápis:	IORWF f, d
Operace:	(W or f) → f, d
Popis:	Provede logický součet (OR) obsahu registrů f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

SUBLW	SUB Literal and W
Zápis:	SUBLW k
Operace:	(k - W) → W
Popis:	Odečte obsah registru W od konstanty k, výsledek uloží do registru W.
Ovlivňuje:	C, DC, Z

SUBWF	SUBtract W from F
Zápis:	SUBWF f, d
Operace:	(f - W) → d
Popis:	Odečte obsah registru W od obsahu registru f, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	C, DC, Z

XORLW	eXclusive OR Literal with W
Zápis:	XORLW k
Operace:	(W xor k) → W
Popis:	Provede nonekvivalneci (XOR) obsah registru W s konstantou k, výsledek uloží do registru W.
Ovlivňuje:	Z

XORWF	eXclusive OR W with F
Zápis:	XORWF f, d
Operace:	(W xor f) → d
Popis:	Provede nonekvivalneci (XOR) obsah registrů f a W, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	Z

A.1. Instrukce nulování a nastavení

BCF	Bit Clear F
Zápis:	BCF f,b
Operace:	0 → f (b)
Popis:	Vynuluje bit b v registru f.
Ovlivňuje:	–

BSF	Bit Set F
Zápis:	BSF f,b
Operace:	1 → f (b)
Popis:	Nastaví do log. 1 bit b v registru f.
Ovlivňuje:	–

CLRF	CLeaR F
Zápis: CLRF f	
Operace:	00h → f
Popis:	Vynuluje obsah registru f.
Ovlivňuje:	Z

CLRW	CLeaR W
Zápis:	CLRW
Operace:	00h → W
Popis:	Vynuluje obsah registru W a nastaví Z bit ve stavovém registru.
Ovlivňuje:	Z

CLRWDT	CLeaR WatchDog Timer
Zápis:	CLRWDT
Operace:	00h → WDT, 0 → WDT předdělič
Popis:	Nuluje čítač WDT a jeho předděličku, je-li k WDT připojená. Nastaví se bity TO a PD.
Ovlivňuje:	1 → TO, 1 → PD

A.2. Instrukce přesunu dat

MOVF	MOVe F
Zápis:	MOVF f,d
Operace:	(f)→ d
Popis:	Obsah registru f přesuneme do registru W (je-li d=0) nebo zpět do registru f (je-li d=1).
Ovlivňuje:	Z

MOVLW	MOVe Literal to W
Zápis:	MOVLW k
Operace:	k $\rightarrow$ W
Popis:	Registr W je naplněn osmibitovou konstantou k.
Ovlivňuje:	–

MOVWF	MOVe W to F
Zápis:	MOVWF f
Operace:	W $\rightarrow$ f
Popis:	Obsah registru W přesuneme do registru f.
Ovlivňuje:	–

RLF	Rotate Left F through carry
Zápis:	RLF f, d
Operace:	$f < n > \rightarrow d < n+1 >, f < 7 > \rightarrow C, C \rightarrow d < 0 >$
Popis:	Rotuje obsah registru f doleva přes bit C (carry), výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	C

RRF	Rotate Right F through carry
Zápis:	RRF f, d
Operace:	$f < n > \rightarrow d < n-1 >, f < 0 > \rightarrow C, C \rightarrow d < 7 >$
Popis:	Rotuje obsah registru f doprava přes bit C (carry), výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	C

SWAPF	SWAP F
Zápis:	SWAPF f, d
Operace:	$f < 0:3 > \rightarrow d < 4:7 >, f < 4:7 > \rightarrow d < 0:3 >$
Popis:	Zamění spodní a horní 4 bity (nibble) obsah registru f, výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1).
Ovlivňuje:	–

A.3. Instrukce podprogramů a přerušení

CALL	subroutine CALL
Zápis:	CALL k
Operace:	$PC + 1 \rightarrow TOS; k \rightarrow PC < 10 : 0 >; PCLATH < 4 : 3 > \rightarrow PC < 12 : 11 >$
Popis:	Návratovou adresu (PC+1) uloží do zásobníku, konstanta k se uloží na spodních 11 bitů PC, zbývající bity PC se doplní z registru PCLATH. Program pokračuje podprogramem na adrese PC.
Ovlivňuje:	–

RETLW	RETurn Literal to W
Zápis:	RETLW k
Operace:	$k \rightarrow W, TOS \rightarrow PC$
Popis:	Návrat z podprogramu. Naplní PC ze zásobníku a registr W naplní konstantou k.
Ovlivňuje:	–

RETURN	RETURN from subroutine
Zápis:	RETURN
Operace:	TOS $\rightarrow$ PC
Popis:	Návrat z podprogramu. Naplní hodnotu PC ze zásobníku.
Ovlivňuje:	–

RETFIE	RETurn From IntErrupt
Zápis:	RETFIE
Operace:	$TOS \rightarrow PC, 1 \rightarrow GIE$
Popis:	Návrat z přerušení. Naplní hodnotu PC ze zásobníku a povolí přerušení nastavením bitu GIE (Global Interrupt Enable) do log. 1.
Ovlivňuje:	–

A.4. Instrukce skoků v programu

BTFSC	Bit Test F, Skip if Clear
Zápis:	BTFSC f, b
Operace:	skok, je-li f (b)=0
Popis:	Je-li bit b v registru f v log. 0, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Ovlivňuje:	–

BTFSS	Bit Test F, Skip if Set
Zápis:	BTFSS f,b
Operace:	skok, je-li f (b)=1
Popis:	Je-li bit b v registru f nastaven na log. 1, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Ovlivňuje:	–

DECFSZ	DECrement F and Skip if Zero
Zápis:	DECFSZ f, d
Operace:	(f - 1) → d, skok, je-li výsledek 0
Popis:	Odečte jedničku od obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Ovlivňuje:	–

GOTO	GO TO address (unconditional jump)
Zápis:	GOTO k
Operace:	$k \rightarrow PC<8:0>, PA2, PA1, PA0 \rightarrow PC<11:9>$
Popis:	Konstanta k (bere se z ní 9 bitů !!!) se uloží na spodních 9 bitů PC, zbývající 3 bity PC se doplní z bitů PA2, PA1 a PA0 v registru STATUS procesoru. Program pokračuje kódem na adrese PC.
Ovlivňuje:	–

INCFSZ	INCrement F and Skip if Zero
Zápis:	INCFSZ f, d
Operace:	(f + 1) → d, skok, je-li výsledek 0
Popis:	Přičte jedničku k obsahu registru f a výsledek uloží do registru W (je-li d=0) nebo do registru f (je-li d=1). Je-li výsledek 0, následující instrukce se neprovede. Jinak program pokračuje na následující instrukci.
Ovlivňuje:	–

A.5. Zvláštní instrukce

NOP - No OPeration	
Zápis:	NOP
Operace:	neprovede nic
Popis:	neprovede nic
Ovlivňuje:	–

SLEEP	SLEEP
Zápis:	SLEEP
Operace:	00h → WDT, 0 → prescaler, 1 → TO, 0 → PD
Popis:	Vynuluje power-down bit PD, nastaví time-out bit TO, vynuluje čítač Watchdog a jeho předděličku. Procesor přejde do stavu SLEEP, oscilátor je vypnut.
Ovlivňuje:	TO, PD

B. Obsah přiloženého CD

`bin/`
: ClickOnce instalátor aplikace SimulatorPIC

`doc/`
: Dokumentace práce ve formátu PDF, včetně všech příloh a zdrojových souborů pro sázecí systém LaTeX.

`src/`
: Kompletní zdrojové texty programu SimulatorPIC..

`readme.txt`
: Instrukce pro instalaci a spuštění programu SimulatorPIC.

Navíc CD/DVD obsahuje:

`literature/`
: Dokumentace mikroprocesoru PIC16F877A ve formátu PDF.