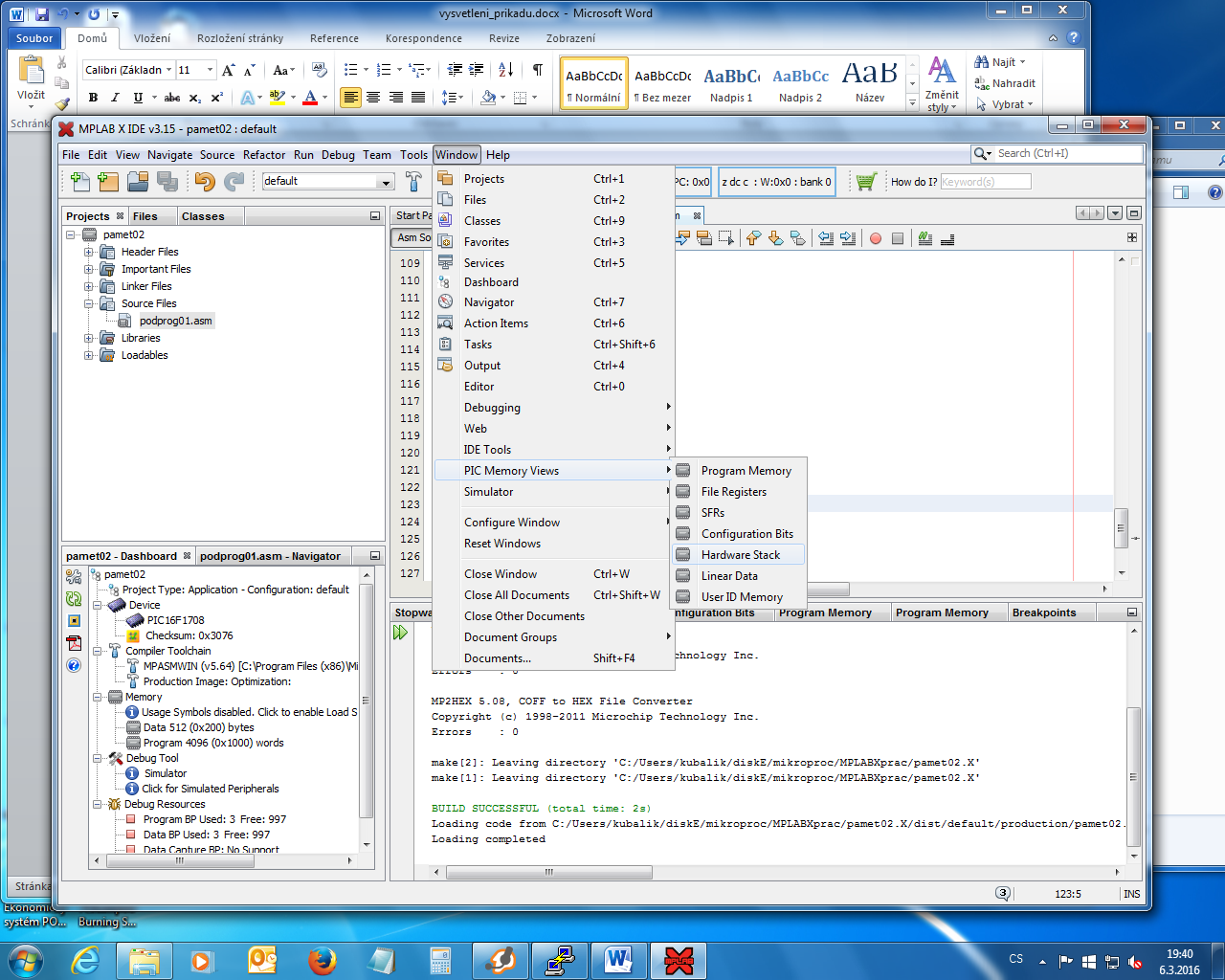
**file podprog01.asm**

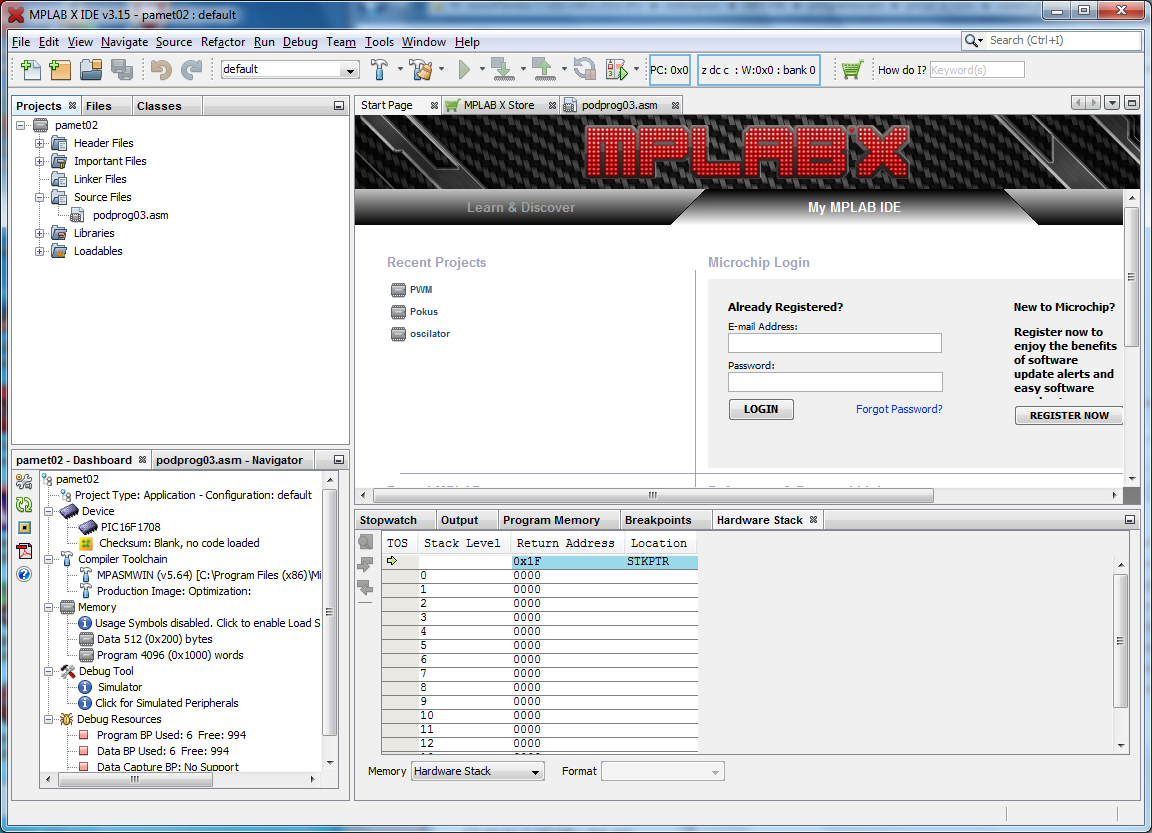
je základní demonstrace chování programu při volání podprogramů. Přeložíme jako Debug, stopneme-Pause, uděláme Reset. Při simulaci musíme používat klávesu F7 – tím simulátor „vleze“ dovnitř simulovaného pdprogramu.

Dále si zapneme „Hardware Stack“

Window – Pic Memory Views –Hardware stack



Okno vypadá následovně:



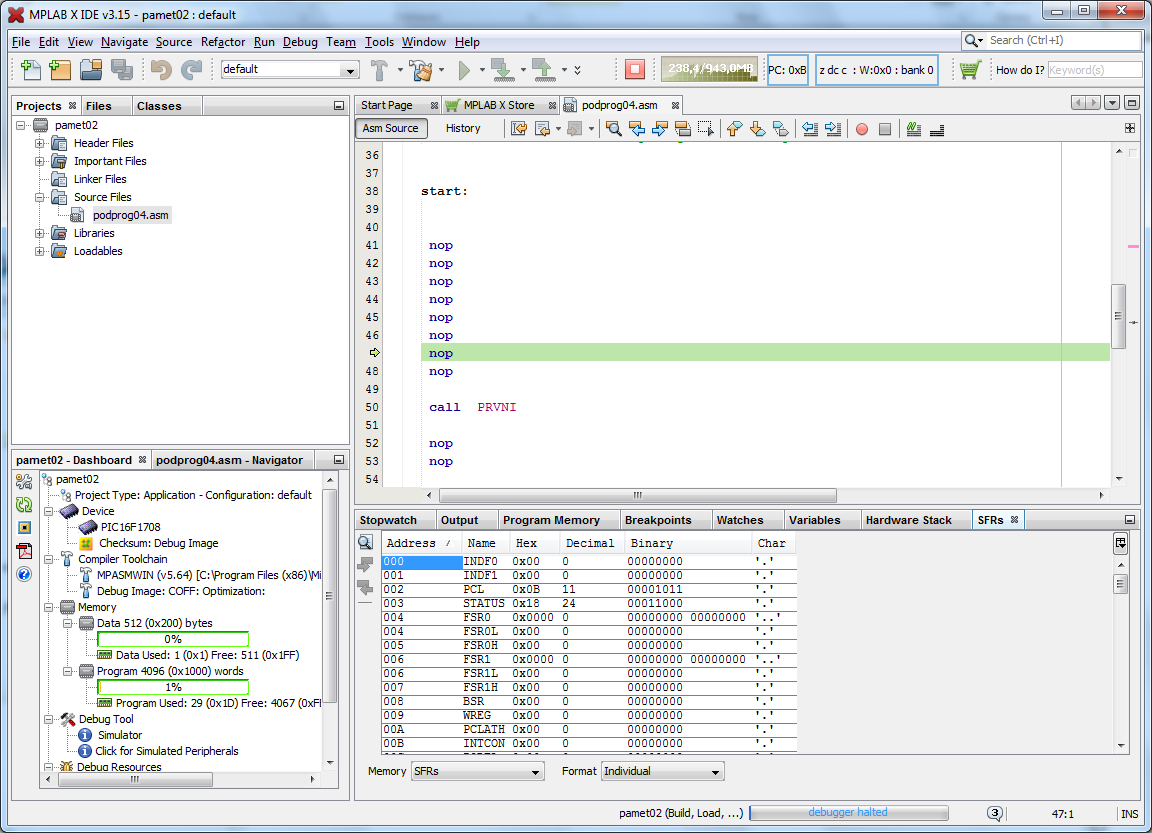
Vidíme v něm 16 buněk pro ukládání návratových adres, na začátku je TOS (Top Of Stack) nastaven na adresu 0x1F. To, znamená, že v pětibitovém registru STKPTR je číslo 0x1F Zdá se to podivné, ale při volání podprogramu se nejdříve udělá INC STKPTR, a teprve potom se uloží návratová adresa. První buňka, do které se bude ukládat návratová adresa, je na adrese 0x00. Zkuste si to – přičtěte 1 k číslu 0x1F, nezapomeňte, že registr je pětibitový.

Dále si stejným způsobem přihodíme Program Memory (tam vidíme adresy jednotlivých instrukcí)

Dále si do Widow – debugging - Watches přidáme registr PC – jako PCL, to je jeho dolních 8 bitů, abychom viděli, co se děje. Také si tam dáme STKPTR

Pak mačkáme F7, díváme se do Stacku, a snažíme se pochopit, co se tam děje – proč se tam objevují která čísla, proč a jak se mění STKPTR a podobně. Celý proces opakujeme pořád dokolečka, až budeme úplně přesně vědět, co se děje při volání podprogramů

Pro správné pochopení toho, co se děje, je životně důležitý obsah PC registru. Ten vidíme v okně simulátoru nahoře



**file podprog02.asm**

Tady je příklad špatné konstrukce volání. Pusťte si to se stejným nastavením simulátoru jako v minulém případě a dívejte se, co to dělá. Ujasněte si, proč je to špatně.

Na konci dojde ke „Stack overflow“ - přetečení STACKu . Procesor na to reaguje tím, že udělá RESET – skáče na adresu 0. (kap. 3.6, strana 42 datasheetu .) . Toto chování se dá vypnout nastavením bitu STVREN v registru CONFIG2 ( strana 51, REGISTER 4-2:) . Je ovšem doufám jasné, že při přetečení STACKu procesor neví, kam se má při RETURN vrátit, takže vypnout tento RESET je pro nás nanic (samozřejmě, Opravdový Programátor si s tím může vyhrát)

**file podprog03.asm**

představuje takzvané rekurentní volání procedur – podprogram volá sám sebe. Je samozřejmé, že musíme počet těchto volání omezit, jinak doje ke Stack Overflow. K tomu slouží buňka POCITATKO, dejte si ji také do Watch.

**file podprog04.asm**

představuje špatně napsané volání za hranicí 2k paměti. Zde je velmi nutné sledovat obsah registru PC, ujasnit si, jak se konstruuje výsledná adresa pro skok, podívat se, co máme v registru PCLATH, a sami si sestavit adresu, na kterou bude program skákat při provedení instrukce CALL. Pokud budete simulovat dost dlouho, dojte ke STACK overflow – zkuste si to, dívejte se do STACKu.

**file podprog05.asm**

představuje dobře napsané volání za hranicí 2k paměti. Do registru PCLATH musíme uložit číslo, které po „zkompletování“ s dolními jedenácti bity registru PC dá adresu za hranicí 0x0800 .

Uděláme to jednoduše. Do tabulky ve file Volani\_podprogramu.doc si do řádku „PC po provedení instrukce“ napíšeme v binárním kódu adresu, kam chceme skákat, tedy 0x0800. No a podíváme se do řádku PCLATH, jaké číslo tam musíme uložit. A je to !

V programu máte nastavení PCLATH dvakrát. Poprvé je to zcela standardní zápis čísla do registru s použitím instrukce movwf. Podruhé je použita instrukce movlp. Na její význam se podíváme do <http://ozeas.sdb.cz/panska/mikroproc/16F1708/programovani/prirucky/instukce.docx>

A samozřejmě si program pustíme v simulátoru, jak jinak.