Politechnika Białostocka

Wydział Elektryczny
Katedra Telekomunikacji i Aparatury Elektronicznej

Instrukcja do zajęć laboratoryjnych z przedmiotu:

Architektura i Programowanie Procesorów Sygnałowych

Kod: TS1C510210

Numer ćwiczenia: 1

Temat ćwiczenia:

Podstawowe procedury obliczeniowe i tryby adresowania procesora sygnałowego TMS320C67x. Arytmetyka stałoprzecinkowa.

Opracował:
dr inż. Dariusz Jańczak

Białystok 2017

Architektura i Programowanie Procesorów Sygnałowych

Temat: Podstawowe procedury obliczeniowe i tryby adresowania procesora sygnałowego TMS320C67x. Arytmetyka stałoprzecinkowa.

1. Cel ćwiczenia

Celem ćwiczenia jest ugruntowanie wiedzy studentów oraz nabycie przez nich umiejętności z zakresu podstaw programowania mikroprocesorów. W ramach zajęć studenci nabywają podstawowe umiejętności z zakresu programowania w języku niskiego poziomu oraz posługiwania się zintegrowanym środowiskiem programistycznym. Studenci zdobywają umiejętność tworzenia, uruchamiania i testowania oprogramowania procesorów na przykładzie procesora sygnałowego (DSP) TMS320C6713. Poznają zasady pisania i uruchamiania programów w asemblerze tego procesora, jego podstawowe instrukcje, tryby adresowania oraz specyficzne funkcje realizowane przez procesory sygnałowe. Realizowanymi zadaniami w bieżącym ćwiczeniu są podstawowe działania arytmetyczne i logiczne z wykorzystaniem działań stałoprzecinkowych. Zadania obejmują obliczenia prowadzone na danych uzyskanych natychmiastowo oraz poprzez adresowanie pośrednie, w tym z modyfikacją adresu i z wykorzystaniem adresowania kołowego.

2. Zagadnienie do opracowania przed przystąpieniem do zajęć

Przed przystąpieniem do zajęć należy opracować następujące zagadnienia:

- podstawowe pojęcia techniki mikroprocesorowej (jak np. ALU, rejestr, akumulator, adres, architektura akumulatorowa i rejestrowa, …)
- stałoprzecinkowe reprezentacje liczb:
  - liczby całkowite NKB i U2 (zinterpretować ciąg bitów o długości 8, 16, 32 i 64 w różnych formatach),
  - liczby ułamkowe ze znakiem – format Qn,
- arytmetyka stałoprzecinkowa,
- działania logiczne na danych o długości 8, 16, 32 bitów (zerowanie, ustawianie, testowanie poszczególnych bitów),
- adresowanie pośrednie, adresowanie pośrednie z modyfikacją adresu, dane natychmiastowe,
- adresowanie liniowe i cyrkularne,
- architektura jednostki centralnej procesora TMS320C6713 (patrz SPRU189),
- realizacja podstawowych działań (dodawanie, odejmowanie, mnożenie, operacje logiczne) - składnia asemblera TMS320C6000 (SPRU186 rozdz. 3.5) w szczególności lista i składnia rozkazów stałoprzecinkowych procesora TMS320C6713 (patrz SPRU189 rozdz. 3),
- zasady inicjalizacji zmiennych różnych typów (dyrektywy: .byte, .int, .long, .word, .half, .short,).

3. Przebieg ćwiczenia

Ćwiczenia laboratoryjne prowadzone są w oparciu o zestaw TMS320C6713 DSP Starter Kit. DSK składa się ze sprzętu i oprogramowania umożliwiającego realizację procedur przetwarzania sygnałów w oparciu o procesor TMS320C6713.

3.1 Tworzenie nowego projektu:

- w katalogu d:/apps/mojprojekt/ utworzyć nowy projekt,
- przegrać do niego zawartość katalogu d:/apps/lab1/,
- odpowiednimi komendami dodać do projektu pliki:lab1.cmd, lab1.asm,
- zapisać projekt i przeanalizować utworzony plik *.pjt,
- ustawić odpowiednie opcje asemblera i linkera niezbędne do kompilacji i uruchomienia programu,
- skompilować i uruchomić przykładowy program,
- przeanalizować możliwość uruchamiania i śledzenia programu (uruchamianie po kroku, pułapki, zawartość rejestrów).

3.2 Dyrektywy asemblera, tryby adresowania, działania arytmetyczne i logiczne:

- przeanalizować strukturę pliku lab1.cmd
- w oparciu o instrukcje asemblera napisać następujące programy:
  - odczytujący z pamięci do rejestru i zapisujący z rejestru do pamięci liczbę 8, 16, 32 i 40 bitową,
  - dodający dwie liczby 16-bitowe i dwie 32-bitowe oraz 32 i 40-bitową (wynik działania zapisać do pamięci),
  - mnożący dwie liczby 16-bitowe (wynik działania zapisać do pamięci),
  - zerujący wybrany rejestr,
  - zerujący bity b3 i b7 oraz ustawiający bity b2 i b24.

3.3 Zadania złożone:

Wykonać zestaw zadań przydzielony grupie przez prowadzącego. Realizacja zadań będzie wymagała znajomości zasad:

- działań arytmetycznych na liczbach różnej długości (8, 16, 32, 40 bitów) w kodzie NKB, U2, Qn,
- operacji logicznych,
- mnożenia liczb długich,
- adresowania pośredniego prostego, z przesunięciem i z modyfikacją adresu,
- adresowania pośredniego cyrkularnego.

4. Sprawozdanie powinno zawierać:

Realizację wszystkich punktów ćwiczenia w tym:

- kody źródłowe wraz z opisem,
- wyniki działania procedur (stan i zmiany wartości rejestrów i komórek pamięci),
- uwagi i wnioski nasuwające się w trakcie wykonywania ćwiczenia.

5. Wymagania BHP

W trakcie realizacji programu ćwiczenia należy przestrzegać zasad omówionych we wstępie do ćwiczeń, zawartych w: „Regulaminie porządkowym w laboratorium" oraz w „Instrukcji obsługi urządzeń elektronicznych znajdujących się w laboratorium z uwzględnieniem przepisów BHP". Regulamin i instrukcja są dostępne w pomieszczeniu laboratoryjnym w widocznym miejscu.

6. Literatura

1. Kowalski H. A., *Procesory DSP dla praktyków,* BTC, Legionowo, 2013
2. Kowalski H. A., *Procesory DSP w przykładach,* BTC, Legionowo, 2012.
3. Dąbrowski A. (red.) *Przetwarzanie Sygnałów Przy użyciu Procesorów Sygnałowych*, Wyd. Politechniki Poznańskiej, Poznań 2000.
4. Gryś S., *Arytmetyka komputerów*, PWN, Warszawa, 2007
5. Texas Instruments, *TMS320C67x DSP CPU and Instruction Set Reference Guide*, 2006.
6. Texas Instruments, *TMS320C6000 Assembly Language Tools User's Guide*, 2008.
7. Texas Instruments, *TMS320C6000 Optimizing Compiler User's Guide*, 2017.
8. Texas Instruments, *TMS320C6000 Programmer's Guide*, 2006.
9. Texas Instruments, *Code Composer Studio Getting Started Guide*, 2014.
10. Welch T. B., Wright C.H.G., Morrow M.G., *Real-time Digital Signal Processing from Matlab to C with the TMS320C6x DSPs,* Taylor & Francis, 2012.

Dodatek 1.

Środowisko Uruchomieniowe CCS – skróty klawiaturowe

F7 - Built

F10 – Step Over

Ctrl+F10 – Run to Cursor

Ctrl+Shift+F10 – Set PC to Cursor

F11 – Step Into

F5 - Run

Ctrl+Shift+F5 - Restart

Alt+F5 - Animate

Ctrl+R – Reset CPU

Dodatek 2.

Dyrektywy Asemblera

Dyrektywy definiujące sekcje

	opis
.data	sekcja danych - inicjalizacja danych
.text	sekcja kodu – programu
.sect *„nazwa_sekcji”*	sekcja o nazwie *“nazwa_sekcji”* - kod

Inicjalizacja stałych

	opis
.byte *wartość1 [, ... , wartośćn]*	inicjalizacja jednego lub więcej kolejnych bajtów
.half *wartość1 [, ... , wartośćn]* **.short** *wartość1 [, ... , wartośćn]*	inicjalizacja półsłów - 16 bitowych liczb całkowitych
.int *wartość1 [, ... , wartośćn]* **.long** *wartość1 [, ... , wartośćn]* **.word** *wartość1 [, ... , wartośćn]*	inicjalizacja słów - 32 bitowych liczb całkowitych

Składnia instrukcji

[label[:]] [\|] [[register]] mnemonic [unit specifier] [operand list] [;comment]

Dodatek 3.

Przykładowy program

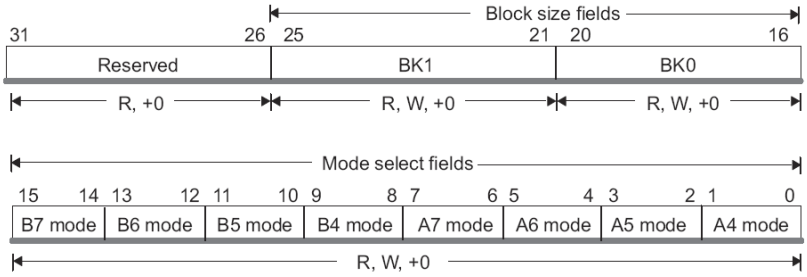
```
                .global start

                .data
arg1:           .int    0x12
arg2:           .int    0x10
wynik: .word    0


                .text
start:          MVK.S1      0x3, A1         ;argument natychmiastowy do rej.A1
                MVK.S1      0xf, A2
                ADD.L1      A1, A2, A3      ;suma do rej.A3
                                            ;
;adresowanie pośrednie
                MVKL.S1     arg1,A0         ;wpis młodszej połowy adresu arg1 do rej.A0
                MVKH.S1     arg1,A0         ;wpis starszej połowy adresu arg1 do rej.A0
                LDW.D1      *A0, A1         ;ładow. arg1 z pamięci do rej.A1 (adresowanie
pośrednie)
                LDW.D1      *++A0, A2       ;ładowanie arg2 do rej.A2 (adr. pośr. z
preinkrement.)
                NOP   4                     ; niezbędne opóźnienie do realizacji LDW
                ADD .L1     A1, A2, A3
                STW   .D1   A3, *+A0[0x1];zapis. zaw. A3 do pam. (adr.pośr. z przesunięc.)
                .end
```

Dodatek 4.

Adresowanie pośrednie kołowe

- stosowany z użyciem rejestrów:A4, A5, A6, A7, B4, B5, B6, B7
- rozmiar definiowany w BK0, BK1 w rejestrze **AMR**
- początek bufora od adresu xx0…0 (N+1 zer)

Addressing Mode Register (AMR)



Legend: R Readable by the **MVC** instruction
W Writeable by the MVC instruction
+0 Value is zero after reset

Addressing Mode Register (AMR) Mode Select Field Encoding

Mode	Description
0 0	Linear modification (default at reset)
0 1	Circular addressing using the BK0 field
1 0	Circular addressing using the BK1 field
1 1	Reserved

Block Size Calculations

N	Block Size	N	Block Size
00000	2	10000	131 072
00001	4	10001	262 144
00010	8	10010	524 288
00011	16	10011	1 048 576
00100	32	10100	2 097 152
00101	64	10101	4 194 304
00110	128	10110	8 388 608
00111	256	10111	16 777 216
01000	512	11000	33 554 432
01001	1 024	11001	67 108 864
01010	2 048	11010	134 217 728
01011	4 096	11011	268 435 456
01100	8 192	11100	536 870 912
01101	16 384	11101	1 073 741 824
01110	32 768	11110	2 147 483 648
01111	65 536	11111	4 294 967 296

Note: When N is 11111, the behavior is identical to linear addressing.