# Raport Laboratorium 2

**Organizacja i Architektura komputerów**

**Autor: Uladzimir Kaviaka (257276)**

**Cel laboratorium:**

Zadanie polegało na napisaniu kalkulatora liczb zmiennoprzecinkowych, przyjęto następujące założenia:

- Wczytywanie/wypisanie realizowane za pomocą **scanf, printf**
- Liczby wpisywane/wypisywane w sposób dziesiętny
- Dodanie trybów zaokrąglania FPU dla wyniku

**Opis algorytmu:**

Wczytujemy dwie podane liczby, wczytujemy wybrany typ operacji, wczytujemy wybrany tryb zaokrąglania. Robimy inicjalizację jednostki FPU, następnie na podstawie trybu zaokrąglania ustawiamy słowo kontrolne dla jednostki FPU, i wykonywamy wybraną operację po czym wyświetlamy wynik

**Implementacja programowa:**

**1. Rezerwacja pamięci pod zmienne tekstowe, zmienne operacji oraz trybu zaokrąglania oraz dla dwóch zmiennych na podstawie których będzie wykonana wybrana operacja.**

```
87      msg1: .ascii "Podaj pierwsza liczbe\n\0"
88      msg2: .ascii "Podaj druga liczbe\n\0"
89      menu: .ascii "Wybierz typ operacji\n1. Dodawanie\n2. Odejmowanie\n3. Dzielenie\n4. Mnozenie\n\0"
90      control: .ascii "Typ zaokraglania\n1. Do zera\n2. W gore\n3. W dol\n4. Do najblizszej\n\0"
91      output: .ascii "Wynik\n\0"
92
93      first_d: .double 1.0
94      second_d: .double 2.0
95      output_d: .double 0.0
96
97      operation: .int 0           #zmienna dla przechowywania wybranej operacji
98      control_operation: .int 0   #zmienna dla przechowywania wybranego zaokraglania
99
100     format_d: .ascii "%lf\0"    #format wypisywania double
101     format_s: .ascii "%s\0"     #format wypisywania string
102     format_i: .ascii "%i\0"     #format wypisywania calkowitej liczby
103
104     endline: .ascii "\n\0"
```

Najpierw deklarujemy zmienne dla naszego menu oraz komunikacji z użytkownikiem (linijki 87-91), następnie rezerwujemy miejsce pod dwie liczby oraz wynik (linijki 93-95), następnie rezerwujemy miejsce pod dwie liczby do przechowywania wybranej operacji oraz trybu zaokrąglania(linijki 97,98). Na końcu deklarujemy zmienne formatu wypisywania/wpisywania wartości które są potrzebne do do funkcji scanf/printf.

**2. Inicjalizacja jednostki FPU**

Inicjalizacja jednostki FPU wykonuje się za pomocą jednego rozkazu: ***finit***

### 3. Wyświetlanie komunikatów oraz pobranie danych wejściowych

```
109         print_s msg1
110         scan_d first_d
111
112         print_s msg2
113         scan_d second_d
114
115         print_s menu
116         scan_i operation
117
118         print_s control
119         scan_i control_operation
```

Wyświetlamy po kolei komunikaty oraz pobieramy dane wejściowe za pomocą funkcji scanf, tutaj sposób użycia scanf/printf jest realizowany za pomocą makro, które napisałem żeby zredukować ilość kodu.

### 4. Sprawdzenie wybranego trybu zaokrąglania

```
121         cmpb $1, control_operation
122         je cut
123         cmpb $2, control_operation
124         je up
125         cmpb $3, control_operation
126         je down
127         cmpb $4, control_operation
128         je nearest
```

Tutaj po wprowadzeniu numeru trybu zaokrąglania ten tryb jest zapisywany w zmiennej **control_operation,** po czym możemy sprawdzić co jest zapisane w tej zmiennej i następnie ustawić potrzebny tryb zaokrąglania.

### 5. Ustawienie trybu zaokrąglania

W FPU tryb zaokrąglania ustawiany w słowie kontrolnym, za tryb zaokrąglania odpowiadają 10 oraz 11 bit słowa kontrolnego które składa się z 16 bitów.
00 – zaokrąglenie do najbliższej
01 – zaokrąglenie w dól (-inf)
10 – zaokrąglenie do góry (+inf)
11 – obcięcie
Więc wprowadziłem cztery zmienne które będą służyć do inicjalizacji słowa kontrolnego w zależności od trybu zaokrąglania

```
3   cut1: .short 0x200 #najblizsze
4   cut2: .short 0x600 #dol
5   cut3: .short 0xA00 #gora
6   cut4: .short 0xE00 #obciecie
```

Na przykładzie trybu zaokrąglenia do najbliższej bity 10 oraz 11 muszą być ustawione na zera, ale tutaj jest 0x200 co równoważne
**0000 0010 0000 0000**
bit zaznaczony na czerwono odpowiada temu że ustawiamy podwójną precyzję dla jednostki FPU.
To samo jest np. dla trybu zaokrąglania w dół:
Tutaj muszą 11 oraz 10 bity ustawione jako 0 oraz 1

**0000 0110 0000 0000**
Analogicznie dla pozostałych trybów zaokrąglania.

### 6. Makra załadowania słowa kontrolnego na podstawie trybu zaokrąglania

```
70  .macro set_round_cut    #zaokraglanie przez obciecie
71      fldcw cut4
72  .endm
73
74  .macro set_round_up     #zaokraglenie do +inf
75      fldcw cut3
76  .endm
77
78  .macro set_round_down   #zaokraglenie do -inf
79      fldcw cut2
80  .endm
81
82  .macro set_round_nearest    #zaokraglenie do najblizszej
83      fldcw cut1
84  .endm
```

Do załadowania słowa kontrolnego do jednostki FPU służy rozkaz **fldcw (FPU load control word)**. Więc tutaj po prostu ładujemy słowo kontrolne do jednostki FPU

### 7. Makra wykonania operacji FPU

```
46  .macro add_d            #makro do wykonania dodawania FPU
47     fldl first_d
48     faddl second_d
49  .endm
50
51  .macro sub_d            #makro do wykonania odejmowania FPU
52      fldl first_d
53      fsubl second_d
54  .endm
55
56  .macro mult_d           #makro do wykonania mnozenia FPU
57      fldl first_d
58      fmull second_d
59  .endm
60
61  .macro div_d            #makro do wykonania dzielenia FPU
62      fldl first_d
63      fdivl second_d
64  .endm
```

Wszystkie operacje (+,-,/,*) mają ten sam algorytm wykonania za pomocą jednostki FPU. Najpierw ładujemy pierwszą zmienną używając rozkaz **fldl,** prefiks l oznacza podwójną precyzję. Następnie wybieramy rozkaz który odpowiada operacji którą chcemy wykonać np. dodawanie **faddl** i podajemy następnie co chcemy dodać czyli naszą drugą liczbę. Ten sam algorytm wykorzystany dla pozostałych operacji.

**8. Wywołanie makro po sprawdzeniu jaki tryb zaokrąglania wybrany**

```
130   cut:
131       set_round_cut
132       je rounding_set
133   up:
134       set_round_up
135       je rounding_set
136   down:
137       set_round_down
138       je rounding_set
139   nearest:
140       set_round_nearest
141       je rounding_set
```

W punkcie 4 tego raportu było sprawdzenie wybranego trybu zaokrąglania i następnie wykonujemy skok do określonego trybu, tutaj podany kod który ilustruje że po wybranym trybie zaokrąglania wywołujemy funkcję która załaduje potrzebne słowo sterujące do jednostki FPU i przechodzimy dalej.

**9. Sprawdzenie wybranej operacji**

```
143   rounding_set:
144       cmpb $1, operation
145       je add
146       cmpb $2, operation
147       je sub
148       cmpb $3, operation
149       je div
150       cmpb $4, operation
151       je mult
152   add:
153       add_d
154       jmp end
155   sub:
156       sub_d
157       jmp end
158   div:
159       div_d
160       jmp end
161   mult:
162       mult_d
163       jmp end
```

Tutaj znów zwykłe sprawdzenie jaką operację wybraliśmy, następnie przechodzimy do potrzebnej metki i wykonujemy operacje opisane makrami w punkcie 7 tego raportu i kończymy działanie kalkulatora.

**10. Zakończenie pracy kalkulatora**

```
165       take_d output_d
166       print_s output
167       print_d output_d
168       endl
```

Tutaj pobieramy z FPU wynik naszej operacji za pomocą rozkazu **fstl**, wyświetlamy komunikat że to jest wynik, wyświetlamy wynik w postaci liczby dziesiętnej i wpisujemy znak nowej linii.

**Wnioski:** Laboratorium pozwoliło zrozumieć w jaki sposób korzystać z jednostki FPU która pozwala robić obliczenia na liczbach zmiennoprzecinkowych, zapoznałem się również ze składnią słowa

kontrolnego jednostki FPU, zrozumiałem jak wykonywać operację za pomocą rozkazów FPU. Również nauczyłem się pisać makra co pozwala redukować ilość kodu.