ORGANIZACJA I ARCHITEKTURA KOMPUTERÓW

LAB 4

12 CZERWCA 2019
ŚRODA, TN 17:05
AUTOR: WOJCIECH KUR
PROWADZĄCY: DR INŻ. PIOTR PATRONIK

Spis treści

1. Treść ćwiczenia....3
1.1. Zakres i program ćwiczenia....3
1.2. Zrealizowane zadania....3
2. Przebieg ćwiczenia....4
2.1. Konstrukcja pliku źródłowego „integrals”....4
2.2. Konstrukcja plików źródłowych „c_sin”....7
3. Podsumowanie....9
4. Literatura....9

1. Treść ćwiczenia

1.1. Zakres i program ćwiczenia

1.1.1. Napisanie programu obliczającego wartość całki oznaczonej funkcji sin(x), log(x).

a) granice zadawane ze standardowego wejścia

b) wynik wypisywany na standardowe wyjście

c) użycie jednostki zmiennoprzecinkowej (rozkazy/rejestry)

1.1.2. Użycie printf/scanf z poziomu kodu w assemblerze.

1.1.3. Zdefiniowanie funkcji my_sin wykorzystującej rozkaz fsin i udostępnienie jej na poziomie interfejsu języka C.

1.1.4. Napisanie programu drukującego bieżący ślad stosu (stack trace).

1.1.5. Zmierzenie czasu wykonania fragmentu kodu z wykorzystaniem rozkazu rdtsc.

1.2. Zrealizowane zadania

1.1.1. Stworzenie programu obliczającego wartość całki oznaczonej funkcji sin(x) wraz z wszystkimi postawionymi wymaganiami.

1.1.2. W trakcie realizacji programu zostały wykorzystane funkcje printf/scanf.

1.1.3 Stworzenie funkcji my_sin i udostępnienie jej z poziomu C. Funkcja zwraca wartość sinusa w danym przez użytkownika punkcie.

2. Przebieg ćwiczenia

2.1. Konstrukcja pliku źródłowego „integrals”

2.1.1 Listing 2: kod źródłowy integrals.s

```
.data
EXIT =                  1
READ =                   3
WRITE =                  4
STDIN =                  0
STDOUT =                   1
SYSCALL =                  0x80

msg1: .string "Choose integral: 1. sin, 2. log:\n"
msg2: .string "Lower limit:\n"
msg3: .string "Upper limit:\n"
msg4: .string "Integral:\n"
notImplError: .string "Not implemented yet.\n"

format: .string "%d"
float_format: .string "%f"
float_print: .string "%f\n"

algorithm: .int 0

.bss
.comm limitUp, 8
.comm limitDown, 8
.comm result, 8

.text
.global _start
_start:
push $msg1
call printf
add $4, %esp
push $algorithm
push $format
call scanf
add $8, %esp
push $msg2
call printf
add $4, %esp
push $limitDown
push $float_format
call scanf
add $8, %esp
push $msg3
call printf
```

```
add $4, %esp
push $limitUp
push $float_format
call scanf
add $8, %esp

cmp $1, algorithm
je alg1
cmp $2, algorithm
je alg2
jmp end

alg1:
fld limitUp
fcos
fld limitDown
fcos

fsubp
fstl result

jmp print
alg2:
push $notImplError
call printf
add $4, %esp
jmp end
print:
   fldl result
   sub $4, %esp
   fstl (%esp)

   push $float_print
   call printf
   add $8, %esp
end:
mov $EXIT, %eax
mov $0, %ebx
int $SYSCALL
```

2.1.2. Opis programu

Program rozpoczyna się od wczytania przedziału całki. Użytkownik jest poproszony także o wybranie funkcji z jakiej ma być wyliczona. Następnie za pomocą jednostki zmiennoprzecinkowej zostaje wyliczona całka z sinusa (sin = - cos). Wynik zostaje odłożony na stos jako double ze względu na ograniczenia funkcji printf.

2.2. Konstrukcja plików źródłowych „c_sin”

2.2.1 Listing 2: kod źródłowy c_sin.s

```
.text
.global my_sin
.type my_sin, @function
my_sin:
push %ebp
mov %esp, %ebp
fld 8(%ebp)
fsin
mov %ebp, %esp
pop %ebp
ret
```

2.2.2 Listing 3: kod źródłowy main.c

```
#include <stdio.h>

extern float my_sin(float arg);

int main()
{
    float argument;
    printf("Podaj argument dla sinusa:\n");
    scanf("%f", &argument);
   printf("Sinus dla %f:\n%f\n", argument, my_sin(argument));
   return 0;
}
```

2.2.3 Listing 4: kod źródłowy Makefile

```
NAME=nazwa_zadania
all:
   as -g --32 ${NAME}.s -o ${NAME}.o
   gcc -g -m32 -o main main.c ${NAME}.o
   rm ${NAME}.o
```

2.2.3. Opis programu

Program składa się z dwóch plików: funkcji assemblerowej oraz głównego programu w C. Funkcja pobiera ze stosu pierwszy argument funkcji my_sin, wykorzystuje go wywołując funkcję fsin oraz zwraca wynik. Dzięki dyrektywie .global linker potrafi wywołać funkcję także w innym pliku.

W pliku main.c została zaimportowana funkcja assemblerowa poprzez „extern”. Jest to znak dla linkera aby wykorzystał funkcję o podanej nazwie. Użytkownik zostaje poproszony o wpisanie argumentu dla sinusa (w radianach), a następnie funkcja zostaje wywołana, wynik zwrócony i wypisany na standardowe wyjście.

3. Podsumowanie

Zadanie umożliwiło poznanie podstawowych rozkazów dla jednostki zmiennoprzecinkowej oraz prostego sposobu na połączenie assemblera i C. Niestety nie udało się zrealizować całki dla logarytmu naturalnego, jednakże wyniosła by ona „x * ln(x) – x”.

4. Literatura

4.1. http://zak.ict.pwr.wroc.pl/materials/architektura/laboratorium%20AK2/wzorzec%20sprawozdania.pdf, wzorzec sprawozdania

4.2. http://jedrzej.ulasiewicz.staff.iiar.pwr.wroc.pl/Architektura-Komputerow/lab/Architektura-63.pdf, laboratorium architektury komputerów – materiały dr Jędrzeja Ułasiewicza

4.3. https://pl.wikibooks.org/wiki/Asembler_x86, teoria oraz prosty program krok po kroku

4.4. http://quantum-mirror.hu/mirrors/pub/gnusavannah/pgubook/ProgrammingGroundUp-1-0-booksize.pdf?fbclid=IwAR0b_yzxqe1Ib9ANvA1BX5r4fFWKh6TarAiws4QtwKpikw2nGNYaYwcYwfI, Programming from the Ground Up

4.5. http://www.cs.umd.edu/~meesh/cmsc311/links/handouts/ia32.pdf

4.6. http://zak.ict.pwr.wroc.pl/materials/architektura/laboratorium%20AK2/Dokumentacja/Intel%20Penium%20IV/IA-32%20Intel%20Architecture%20Software%20Developers%20Manual%20vol.%202%20-%20Instruction%20Set%20Reference.pdf, dokumentacja intela (Instruction Reference vol.2)