Wrocław, 11.03.2018

Adam Karczewski, 235039
WT-N-7

prowadzący: Aleksandra Postawka

Laboratorium Architektury Komputerów

(0) Podstawy uruchamiania programów asemblerowych na platformie Linux/x86

0.1 Treść ćwiczenia

Zakres i program ćwiczenia:
Nauka podstaw tworzenia i uruchamiania programów napisanych w języku asemblera AT&T(Linux/x86).

Zrealizowanie zadania:
Utworzenie w środowisku 64- bitowym programu „Hello world", wykorzystującego funkcję systemowe: write i exit.
Kompilacja i uruchomienie programu.
Utworzenie pliku makefile i kompilacja z użyciem komendy make.

0.2 Przebieg ćwiczenia

0.2.1 Budowa kodu źródłowego

Program rozpoczyna sekcja danych

```
.data
SYSREAD = 0
SYSWRITE = 1
SYSEXIT = 60
STDOUT = 1
STDIN = 0
EXIT_SUCCESS = 0

buf: .ascii "Hello, world! \n"
buf_len=  .-buf
```

W sekcji .data deklarujemy zmienne, są to kody funkcji systemowych: SYSREAD, SYSWRITE oraz SYSEXIT, dodatkowo deklarujemy konieczne parametry dla tych funkcji. Deklarujemy także bufor zwierający napis zakodowany w ASCII. Obliczmy

jego długość odejmując od bieżącej pozycji w pamięci (.) adres pierwszego znaku w napisie.

```
.text
.globl _start

_start:
movq $SYSWRITE, %rax
movq $STDOUT, %rdi
movq $buf, %rsi
movq $buf_len, %rdx
syscall
```

W sekcji .text zapisujemy algorytm wykonania programu, dyrektywa .globl _start rozpoczyna wykonanie programu. Za pomocą mnemonika movq umieszczamy w kolejnych rejestrach 64b:

1) RAX- kod funkcji SYSWRITE=1
2) RDI- parametr pierwszy- domyślny strumień wyjścia STDOUT=1
3) RSI- bufor do wypisania
4) RDX- długość buforu

Po umieszczeniu w rejestrach odpowiednich parametrów polecaniem syscall wywołujemy przerwanie na procesorze i funkcja zostanie wykonana.

```
movq $SYSEXIT, %rax
movq $EXIT_SUCCESS, %rdi
syscall
```

Na koniec umieszczamy kod funkcji SYSEXIT z parametrem 0.

0.2.2 Kompilacja i uruchomienie programu

Pierwszym etapem jest stworzenie pliku o rozszerzeniu .o w tym celu używamy kompilatora as uruchamiając go poleceniem:

```
as -o zad0.o zad0.s
```

Gdzie zad0.s to plik źródłowy. Następnie konsolidujemy plik. o do pliku wykonywalnego poleceniem:

```
ld-o zad0 zad0.o
```

Program uruchamiamy polecaniem `ścieżka_dostępu/nazwa_pliku`

0.2.3 Kompilacja komendą make

Kompilację programu można usprawnić tworząc plik makefile o konstrukcji:

```
zad0:[\tab]  zad0.o
     [\tab]  ld-o zad0 zad0.o
zad0.o:[\tab] zad0.s
       [\tab] as -o zad0.o zad0.s
```

W pliku tym określamy od końca jaki plik ma powstać, na jakiej podstawie i przy pomocy jakiego polecania. Po stworzeniu powyższego pliku kompilacja zostaje skrócona do polecania `make zad0.s`

0.3 Wnioski

Na laboratorium udało się uruchomić program Hello, world.