lab_8 - Instrukcja do ćwiczenia

Teoria:

Praktyka (lab_8.c i pozostałe pliki):

Działania:

1. Tworzymy bibliotekę w wersji statycznej:

```
gcc -c -o  gcd_static.o  gcd_static.s
gcc -c -o  print_rsp_static.o   print_rsp_static.s
ar  rcs  libstat.a  gcd_static.o  print_rsp_static.o
```

2. Budujemy kod wykonywalny wykorzystując stworzoną bibliotekę:

```
gcc  -no-pie  -o  lab_8_static   lab_8.c   -L.  -lstat
```

3. Uruchamiamy program podając jednocześnie dwie liczby będące argumentami funkcji **gcd_c** i **gcd_a**:

```
     ./lab_8_static    4084   1024
```

4. Program powinien zadziałać prawidłowo – obie funkcje powinny zwrócić wartość **4**, dodatkowym efektem jest wyświetlenie informacji o wartości wskaźnika stosu **%rsp** i wartości zmiennej **counter** (poziomu zagnieżdżenia funkcji **gcd_a**):

```
buba@buba-pc:~/AK/l8$ ./lab_8_static 4084 1024
rsp=7fff4ef49bd0 call,  counter = 1
GCD_c(4084, 1024) = 4
rsp=7fff4ef49bb8 call,  counter = 2
rsp=7fff4ef49bb0 call,  counter = 3
rsp=7fff4ef49ba8 call,  counter = 4
rsp=7fff4ef49ba0 call,  counter = 5
rsp=7fff4ef49b98 call,  counter = 6
rsp=7fff4ef49ba0 ret,   counter = 6
rsp=7fff4ef49ba8 ret,   counter = 5
rsp=7fff4ef49bb0 ret,   counter = 4
rsp=7fff4ef49bb8 ret,   counter = 3
rsp=7fff4ef49bc0 ret,   counter = 2
GCD_a(4084, 1024) = 4
rsp=7fff4ef49bd0 ret,   counter = 1
```

5. Każde wywołanie funkcji **gcd_a** powoduje odłożenie na stosie adresu powrotu (**8** bajtów), więc wartość wskaźnika stosu **%rsp** maleje o **8** bajtów (stos rośnie) –powrót

z funkcji wiąże się z pobraniem ze stosu adresu powrotu, więc wartość wskaźnika stosu **%rsp** rośnie o **8** bajtów (stos maleje).

6. Tworzymy bibliotekę w wersji dynamicznej:

```
gcc -fPIC -c -o  gcd_dynamic.o   gcd_dynamic.s
gcc -fPIC -c -o  print_rsp_dynamic.o  print_rsp_dynamic.s
gcc -shared -o libdyn.so gcd_dynamic.o print_rsp_dynamic.o
```

7. Budujemy kod wykonywalny wykorzystując stworzoną bibliotekę:

```
gcc -fPIC -o lab_8_dynamic  lab_8.c  -L.  -ldyn
```

8. Sprawdzamy jego działanie:

```
./lab_8_dynamic 4084  1024
```

9. Program powinien zadziałać tak samo jak wersja statyczna – różnice mogą się pojawić w wartościach wskaźnika stosu **%rsp**.
10. Modyfikujemy zawartości plików **gcd_static.s** oraz **gcd_dynamic.s** – celem jest wyświetlenie wartości argumentów funkcji **gcd_a**.
11. W obu plikach usuwamy znaki komentarzy (**#**).
12. Tworzymy bibliotekę w wersji statycznej:

```
gcc -c -o  gcd_static.o  gcd_static.s
gcc -c -o  print_rsp_static.o   print_rsp_static.s
ar  rcs  libstat.a  gcd_static.o  print_rsp_static.o
```

13. Budujemy kod wykonywalny wykorzystując stworzoną bibliotekę:

```
gcc  -no-pie  -o  lab_8_static   lab_8.c   -L.  -lstat
```

14. Uruchamiamy program podając jednocześnie dwie liczby będące argumentami funkcji **gcd_c** i **gcd_a**:

```
     ./lab_8_static    4084   1024
```

15. Program powinien zadziałać prawidłowo – dodatkowym efektem jest wyświetlenie informacji o argumentach funkcji **gcd_a**:

```
buba@buba-pc:~/AK/l8$ ./lab_8_static 4084 1024
rsp=7ffcee78cc30 call,  counter = 1
GCD_c(4084, 1024) = 4
rsp=7ffcee78cc18 call,  counter = 2
Called GCD_A(4084,1024)
rsp=7ffcee78cc10 call,  counter = 3
Called GCD_A(1024,1012)
rsp=7ffcee78cc08 call,  counter = 4
Called GCD_A(1012,12)
rsp=7ffcee78cc00 call,  counter = 5
Called GCD_A(12,4)
```

```
rsp=7ffcee78cbf8 call,  counter = 6
Called GCD_A(4,0)
rsp=7ffcee78cc00 ret,   counter = 6
rsp=7ffcee78cc08 ret,   counter = 5
rsp=7ffcee78cc10 ret,   counter = 4
rsp=7ffcee78cc18 ret,   counter = 3
rsp=7ffcee78cc20 ret,   counter = 2
GCD_a(4084, 1024) = 4
rsp=7ffcee78cc30 ret,   counter = 1
buba@buba-PC:~/AK/l8$
```

16. Tworzymy bibliotekę w wersji dynamicznej:

```
gcc -fPIC -c -o  gcd_dynamic.o   gcd_dynamic.s
gcc -fPIC -c -o  print_rsp_dynamic.o  print_rsp_dynamic.s
gcc -shared -o libdyn.so gcd_dynamic.o print_rsp_dynamic.o
```

17. Budujemy kod wykonywalny wykorzystując stworzoną bibliotekę:

```
gcc -fPIC -o lab_8_dynamic  lab_8.c  -L.  -ldyn
```

18. Sprawdzamy jego działanie:

```
./lab_8_dynamic 4084  1024
```

19. Program powinien zadziałać tak samo jak wersja statyczna – różnice mogą się pojawić w wartościach wskaźnika stosu **%rsp**.
20. Modyfikujemy zawartości plików **print_rsp_static.s** oraz **print_rsp_dynamic.s** – celem jest wyświetlenie adresu zmiennej **counter**.
21. W obu plikach usuwamy znaki komentarzy (**#**).
22. Ponownie tworzymy bibliotekę w wersji statycznej:

```
gcc -c -o  gcd_static.o  gcd_static.s
gcc -c -o  print_rsp_static.o   print_rsp_static.s
ar  rcs  libstat.a  gcd_static.o  print_rsp_static.o
```

23. Budujemy kod wykonywalny wykorzystując stworzoną bibliotekę:

```
gcc  -no-pie  -o  lab_8_static   lab_8.c   -L.  -lstat
```

24. Uruchamiamy program podając jednocześnie dwie liczby będące argumentami funkcji **gcd_c** i **gcd_a**:

```
      ./lab_8_static    4084   1024
```

25. Program powinien zadziałać prawidłowo – dodatkowym efektem jest wyświetlenie informacji o adresie zmiennej **counter**:

```
buba@buba-pc:~/AK/l8$ ./lab_8_static 4084 1024
```

```
rsp=7ffd27ed1fc0 call,  counter = 1
&counter=40404d call
GCD_c(4084, 1024) = 4
rsp=7ffd27ed1fa8 call,  counter = 2
&counter=40404d call
Called GCD_A(4084,1024)
rsp=7ffd27ed1fa0 call,  counter = 3
&counter=40404d call
Called GCD_A(1024,1012)
rsp=7ffd27ed1f98 call,  counter = 4
&counter=40404d call
Called GCD_A(1012,12)
rsp=7ffd27ed1f90 call,  counter = 5
&counter=40404d call
Called GCD_A(12,4)
rsp=7ffd27ed1f88 call,  counter = 6
&counter=40404d call
Called GCD_A(4,0)
rsp=7ffd27ed1f90 ret,   counter = 6
&counter=40404d ret
rsp=7ffd27ed1f98 ret,   counter = 5
&counter=40404d ret
rsp=7ffd27ed1fa0 ret,   counter = 4
&counter=40404d ret
rsp=7ffd27ed1fa8 ret,   counter = 3
&counter=40404d ret
rsp=7ffd27ed1fb0 ret,   counter = 2
&counter=40404d ret
GCD_a(4084, 1024) = 4
rsp=7ffd27ed1fc0 ret,   counter = 1
&counter=40404d ret
buba@buba-PC:~/AK/l8$
```

26. Tworzymy bibliotekę w wersji dynamicznej:

```
gcc -fPIC -c -o  gcd_dynamic.o   gcd_dynamic.s
gcc -fPIC -c -o  print_rsp_dynamic.o  print_rsp_dynamic.s
gcc -shared -o libdyn.so gcd_dynamic.o print_rsp_dynamic.o
```

27. Budujemy kod wykonywalny wykorzystując stworzoną bibliotekę:

```
gcc -fPIC -o lab_8_dynamic  lab_8.c  -L.  -ldyn
```

28. Sprawdzamy jego działanie:

```
./lab_8_dynamic 4084  1024
```

29. Program powinien zadziałać tak samo jak wersja statyczna – różnice mogą się pojawić w wartościach wskaźnika stosu **%rsp** oraz w adresie zmiennej **counter**:

```
buba@buba-pc:~/AK/l8$ ./lab_8_dynamic 4084 1024
rsp=7ffdcbc56da0 call,  counter = 1
&counter=7ff80809a055 call
GCD_c(4084, 1024) = 4
rsp=7ffdcbc56d88 call,  counter = 2
&counter=7ff80809a055 call
Called GCD_A(4084,1024)
rsp=7ffdcbc56d80 call,  counter = 3
&counter=7ff80809a055 call
Called GCD_A(1024,1012)
rsp=7ffdcbc56d78 call,  counter = 4
&counter=7ff80809a055 call
Called GCD_A(1012,12)
rsp=7ffdcbc56d70 call,  counter = 5
&counter=7ff80809a055 call
Called GCD_A(12,4)
rsp=7ffdcbc56d68 call,  counter = 6
&counter=7ff80809a055 call
Called GCD_A(4,0)
rsp=7ffdcbc56d70 ret,   counter = 6
&counter=7ff80809a055 ret
rsp=7ffdcbc56d78 ret,   counter = 5
&counter=7ff80809a055 ret
rsp=7ffdcbc56d80 ret,   counter = 4
&counter=7ff80809a055 ret
rsp=7ffdcbc56d88 ret,   counter = 3
&counter=7ff80809a055 ret
rsp=7ffdcbc56d90 ret,   counter = 2
&counter=7ff80809a055 ret
GCD_a(4084, 1024) = 4
rsp=7ffdcbc56da0 ret,   counter = 1
&counter=7ff80809a055 ret
buba@buba-PC:~/AK/l8$
```

30. Tworzymy zbiory informacji o symbolach zawartych w wykorzystywanych bibliotekach i plikach wykonywalnych:

```
objdump  -D  lab_8_static  >  lab_8_static.txt
objdump  -D  libstat.a  >  libstat.txt
objdump  -D  lab_8_dynamic  >  lab_8_dynamic.txt
objdump  -D  libdyn.so  >  libdyn.txt
```

31. W stworzonych plikach wyszukujemy symbole **<gcd_a>**, **<print_call_rsp>** oraz **<counter>**.
32. Biblioteka statyczna nie zawiera żadnych informacji o rzeczywistych adresach:

```
0000000000000000 <gcd_a>:
0:    57                          push   %rdi
```

```
1:    56                            push   %rsi
2:    e8 00 00 00 00                callq  7 <gcd_a+0x7>
0000000000000000 <print_call_rsp>:
0:    8b 14 25 00 00 00 00          mov    0x0,%edx
7:    ff c2                         inc    %edx
0000000000000000 <counter>:
0:    00 00                         add    %al,(%rax)
```

33. Program wykonywalny zawiera informacje o rzeczywistych adresach (porównaj adres zmiennej **counter** wyświetlony w pkt. 25):

```
0000000000401233 <gcd_a>:
  401233:   57                      push   %rdi
  401234:   56                      push   %rsi
  401235:   e8 3b 00 00 00          callq  401275 <print_call_rsp>
0000000000401275 <print_call_rsp>:
  401275:   8b 14 25 4d 40 40 00    mov    0x40404d,%edx
  40127c:   ff c2                   inc    %edx
000000000040404d <counter>:
  40404d:   00 00                   add    %al,(%rax)
```

34. Biblioteka dynamiczna zawiera informacje o adresach, ale nie są to rzeczywiste adresy (pkt. **29**) tylko względne (przesunięcia względem ustalonego układu odniesienia):

```
0000000000001139 <gcd_a>:
 1139:    57                      push   %rdi
 113a:    56                      push   %rsi
 113b:    e8 20 ff ff ff          callq  1060 <print_call_rsp@plt>
000000000000117b <print_call_rsp>:
 117b:    48 8d 15 d3 2e 00 00    lea    0x2ed3(%rip),%rdx # 4055 <counter>
 1182:    ff 02                   incl   (%rdx)
0000000000004055 <counter>:
 4055:    00 00                   add    %al,(%rax)
```

35. Program wykonywalny w wersji dynamicznej nie zawiera żadnych informacji o symbolach, które są dołączane dynamicznie (znajdują się w dołączanej bibliotece).
36. Pozostały jeszcze do realizacji kilka eksperymentów i wyciągnięcie z nich wniosków – to już we własnym zakresie:
    a. uruchomienie kilkukrotnie programu z tymi samymi argumentami dla GCD/NWD w wersji z biblioteką statyczną i dynamiczną,
    b. obserwacja zachowania wierzchołka stosu (adresu w %rsp), adresu licznika **counter**, liczby iteracji i wyniku,
    c. zmiana argumentów do liczenia GCD/NWD i powtórzenie wcześniejszych eksperymentów.