Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Science
and Technology

Binutils, Biblioteki statyczne i dynamiczne

Robert Gałat

Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza im. Stanisława Staszica

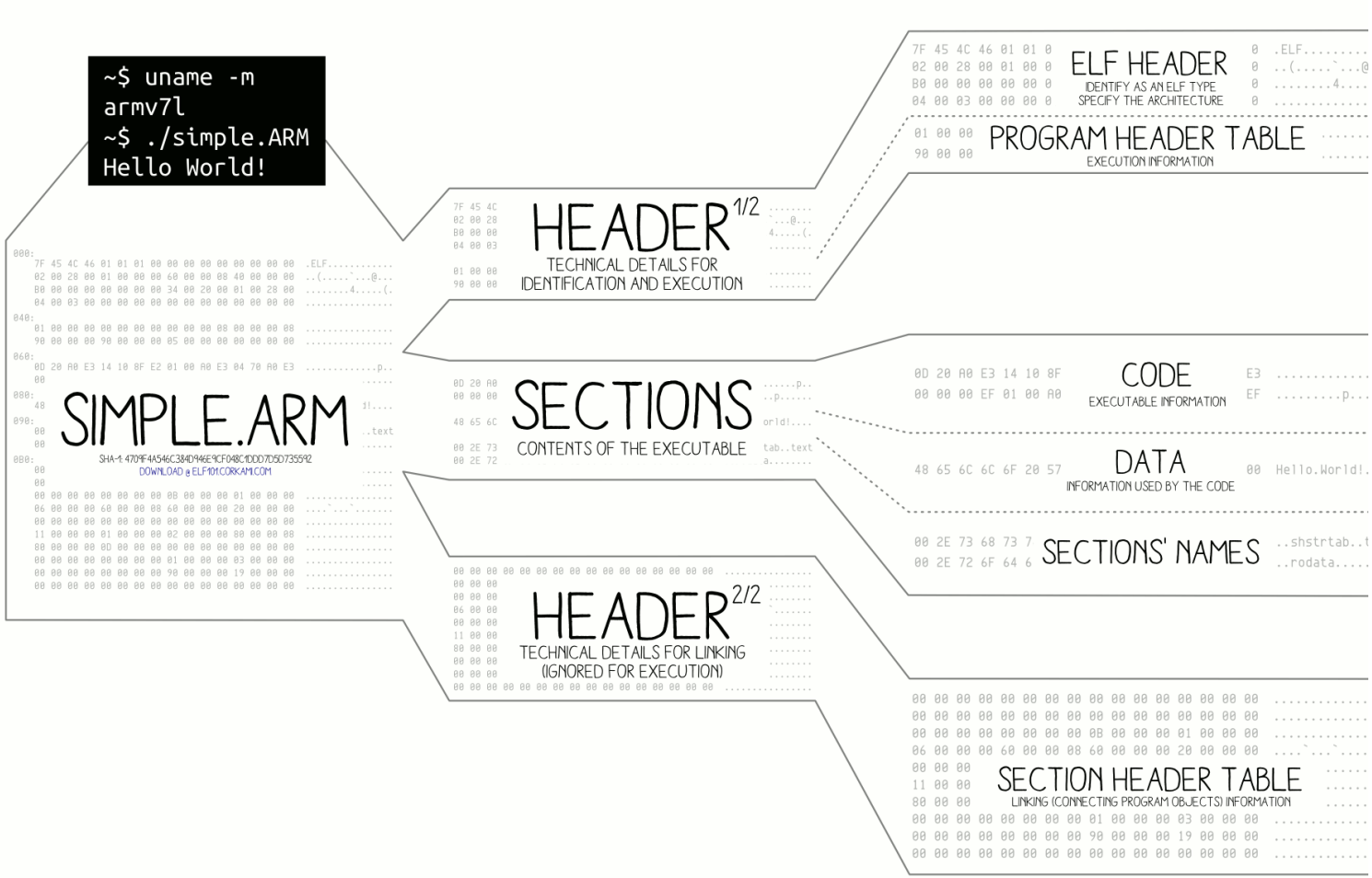
10 IV 2018

1. pliki obiektowe
   - format plików obiektowych
   - Rodzaje plików obiektowych
     - Klasy symboli (nm)

2. Biblioteki
   - Realocation and PIC
   - GOT
   - PLT
   - Dynamiczne ładowanie bibliotek

www.agh.edu.pl

Nagłówki ELF



Rysunek: budowa pliku ELF [4]

ELF

[7]Najważniejsze sekcje zawierają:

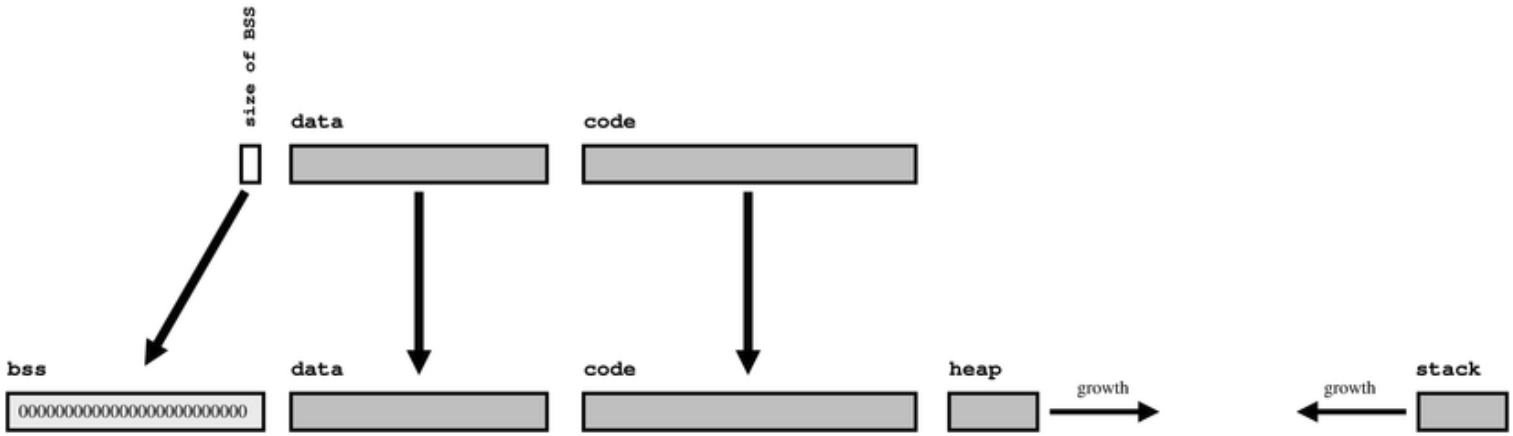
- .bss — niezainicjowane zmienne, sekcja **.bss** zajmuje nieznaczne ilości pamięci, ponieważ zawiera jedynie informacje o zmiennych, a nie przechowuje ich wartości
- .text — wykonywalne części programu
- .data — zainicjowane wartości

Sekcja vs Segment

Sekcje występują przed procesem linkowania i dzielimy je na:

- "raw data" – np. .text, .data itd ...
- "metadata" – np. .symtab, .strtab itd ...

sekcje zawierające metadane nie są konieczne do uruchomienia programu, ale stanową źródło informacji dla debugerów, i programów analizujących pliki binarne [`przyklady/strip`]
Segmenty występują po procesie linkowania i zawierają informacje jak system operacyjny ma je załadować do pamięci



Rysunek: Proces ładowania programu do pamięci [3]

- Pliki przesuwalne (Relocatable)
- Pliki wykonywalne (Executable)
- Pliki współdzielone (Shared Objects)

Przesuwalne (relocatable) pliki obiektowe

Przesuwalne pliki obiektowe to takie których funkcje i zmienne nie są przywiązane do adresu, ale do symboli. [5]

Plik źródłowy:

```
int add(int a, int b)
{
        return a+b;
}
```

[przyklady/00_relocatable]

Wybrane klasy symboli w pliku obiektowym [1]

- B - symbol jest w sekcji BSS (niezainicjowany)
- C - Wspólny symbol, nie zainicjowane
- D - zainicjowane symbol
- R - tylko do odczytu
- T - symbol jest w sekcji text (code)
- U - symbol niezdefiniowany

Przykład użycia programu nm na pliku obiektowym:

Wartość	Klasa	Nazwa
0000000000000000	D	global_initialized_variable
0000000000000004	C	global_variable
0000000000000013	t	local_function
0000000000000021	T	main
0000000000000000	T	not_implemented_function

Wykonywalne (executable) pliki obiektowe

```
int add(int a, int b);

int main()
{
        return add(1,-1);
}
```

[przyklady/01_executable]
binarny plik wykonywalny zawiera adresy, natomiast przenośny plik obiektowy wszystkie adresy ma ustawione na 0

Biblioteki statyczne

Biblioteka statyczna to archiwum plików “relocatable”, więc korzysta się z niej analogicznie jak z tych plików. Nazwa takiej biblioteki zazwyczaj rozpoczyna się od 'lib' a kończy się rozszerzeniem “.a”.

Aby zbudować bibliotekę statyczną korzystamy z polecenia

```
gcc -static -o libfoo.a foo.o
```

Aby skorzystać z takiej biblioteki korzystamy z polecenia

```
gcc example.o -lfoo
```

kolejność linkowania

File	a.o	b.o	libx.a			liby.a		
Object	a.o	b.o	x1.o	x2.o	x3.o	y1.o	y2.o	y3.o
Definitions	a1, a2, a3	b1, b2	x11, x12, x13	x21, x22, x23	x31, x32	y11, y12	y21, y22	y31, y32
Undefined references	b2, x12	a3, y22	x23, y12	y11		y21		x31

Rysunek: zależności przy linkowaniu [3]

Współdzielone (shared)

Aby stworzyć plik biblioteki współdzielonej musimy wywołać następującą komendę:

```
gcc -shared -o libfoo.so foo.o
```

jednak plik obiektowy foo.o musi być odpowiednio skompilowany (flaga -fPIC) ponieważ pliki typu .so muszą być „position independent”

[`przyklady/02_shared`]

Realocation vs Position Independent Code

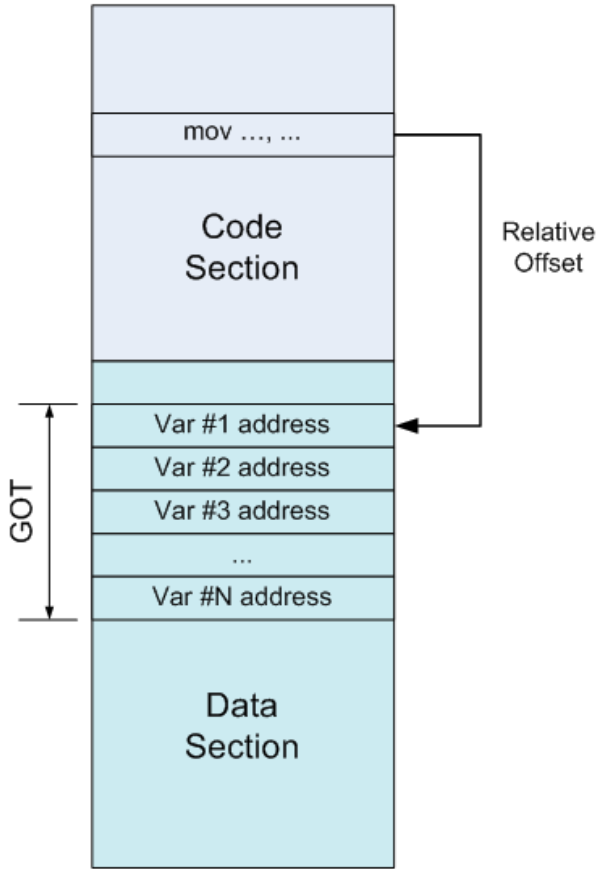
Re alokacja

polega na wyliczeniu poprawnych adresów obiektów z biblioteki współdzielonej przed uruchomieniem programu.

Position Independent Code

jest nowszą technologią która zakłada że poprzez dodanie warstwy abstrakcji sekcja .text ma uprawnienia tylko do odczytu, co zapewnia proste mapowanie adresów, zwiększa bezpieczeństwo i zapewnia współdzielenie kodu biblioteki przez wiele procesów

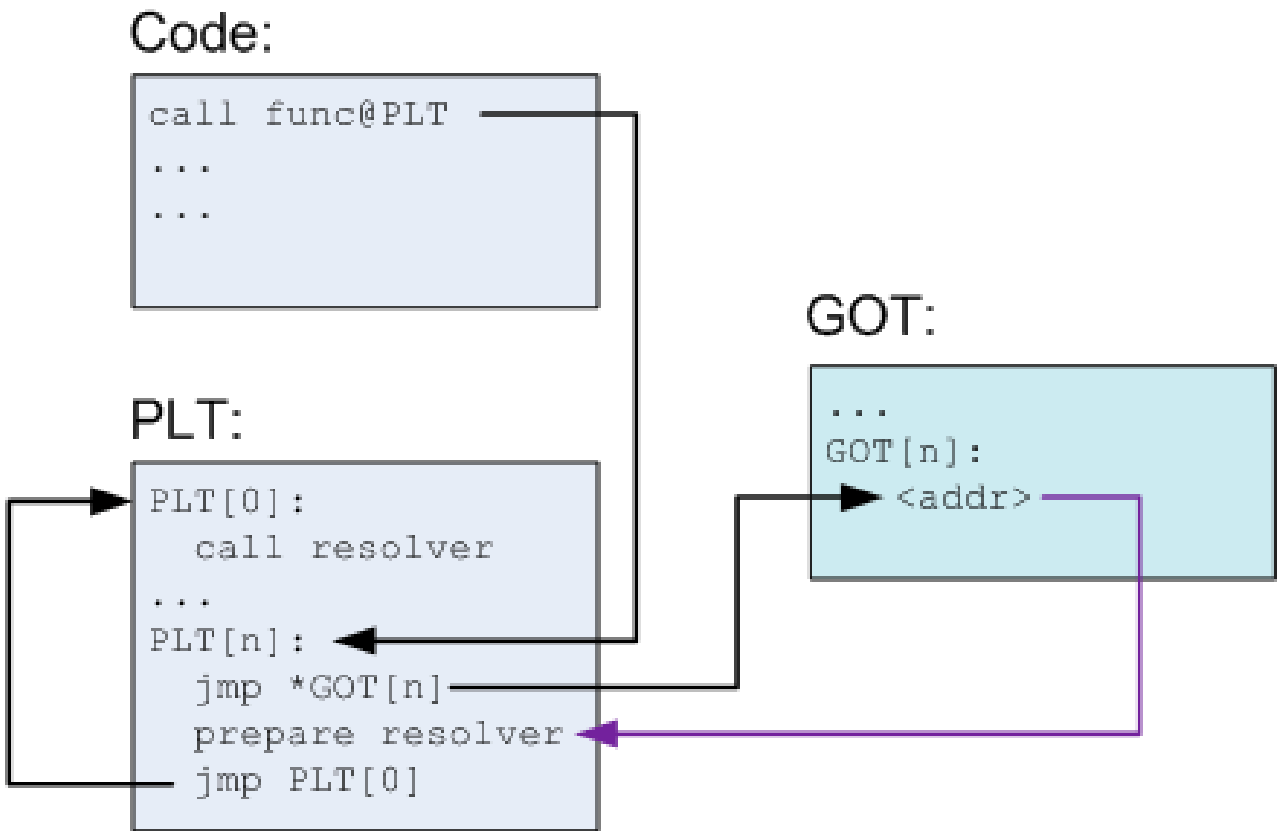
Global Offset Table



Rysunek: Schemat tablicy GOT [2]

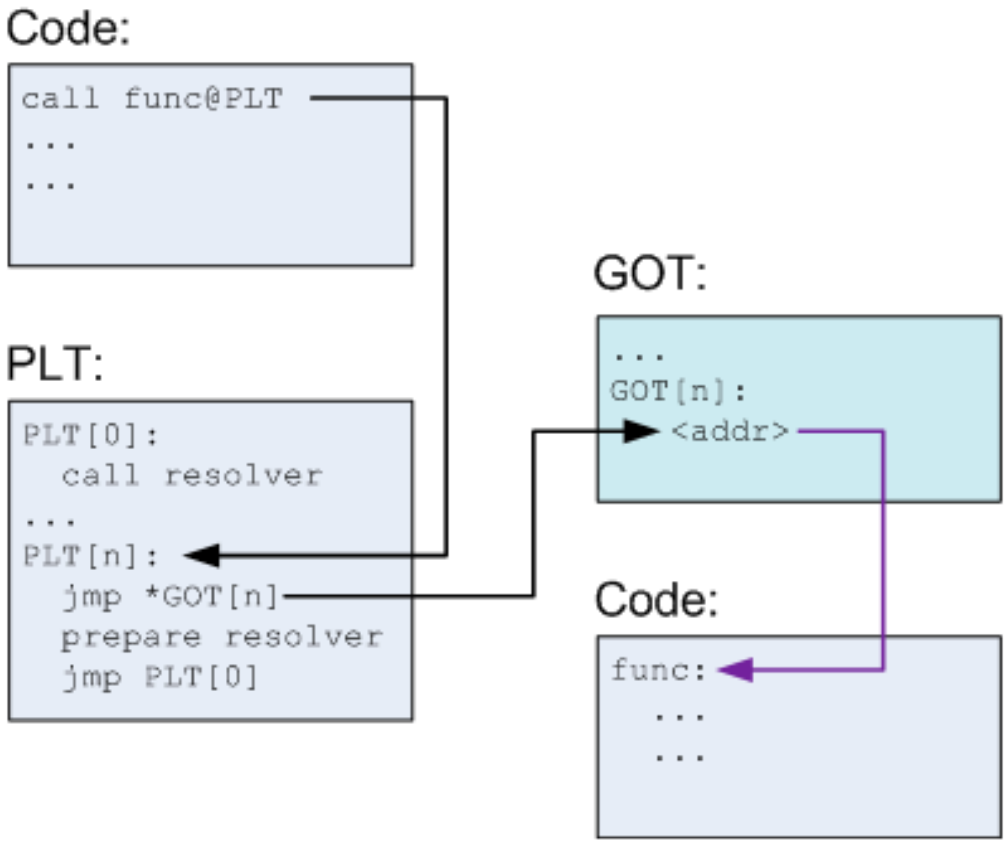
[przyklady/PIC_variable/]

Procedure Link Table



Rysunek: Schemat tablicy PLT przed pierwszym wywołaniem funkcji [2]

Procedure Link Table



Rysunek: Schemat tablicy PLT po pierwszym wywołaniem funkcji [2]

[przyklady/PIC_functions]

dlfcn.h

[6] Biblioteka “dlfcn.h” umożliwia dynamiczne ładowanie bibliotek współdzielonych w trakcie działania programu
Do obiektów biblioteki mamy dostęp poprzez handler zwracany z funkcji `void * dlopen(const char *filename, int flag);`
Do pierwszego argumentu można podać ścieżkę do pliku biblioteki współdzielonej, lub jej nazwę (wtedy będzie szukana w domyślnych katalogach lub w LD_LIBRARY_PATH)
Drugi argument określa czy linkowanie będzie leniwe, czy od razu z linkować wszystkie obiekty.

```
void * dlsym(void *handle, char *symbol);
```

Powyższa funkcja zwraca wskaźnik na obiekt którego żądamy w argumencie
[`przyklady/DL_library`]


Manual programu nm.


Eli Bendersky.
Position independent code (pic) in shared libraries.
[online].
[dostęp: 2018-04-04].


David Drysdale.
Beginner's guide to linkers.
[online], Kwiecień 2018.
[dostęp: 2018-04-04].


Ciro Santilli.
Elf hello world tutorial.
[online].
[dostęp: 2018-04-04].


Carson Tang.
Guide to object file linking.
[online], June 2013.

[dostęp: 2018-04-04].

David A. Wheeler.
Program library howto.
[online], April 2013.
[dostęp: 2018-04-04].

Eric Youngdale.
The elf object file format: Introduction.
[online], April 1995.
[dostęp: 2018-04-04].