ELF1 7D Relocs in i386 - ELF Study 1999

Young W. Lim

2020-04-13 Mon

Outline

1. Based on
2. PIC relocs
   - TOC
   - Two syntactic constructs
   - GOT / PLT based relocs in object files
   - Transformed relocs in shared libraries or executable files
   - Summary
3. Relocs Summary in i386
   - TOC
   - 0. Background
   - 1. Relocs in .o files for executables
   - 2. Relocs in .o files for shared libraries
   - 3. Relocs in executable files
   - 4. Relocs in shared librariy files

Based on

"Study of ELF loading and relocs", 1999
http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Compling 32-bit program on 64-bit gcc

- `gcc -v`
- `gcc -m32 t.c`
- `sudo apt-get install gcc-multilib`
- `sudo apt-get install g++-multilib`
- `gcc-multilib`
- `g++-multilib`
- `gcc -m32`
- `objdump -m i386`

TOC: PIC relocs

1. Two syntactic constructs
2. Reloc sections
3. GOT / PLT based relocs
   R_386_GOT32, R_386_GOTOFF, R_386_PLT32, R_386_GOTPC
4. Transformed relocs
   R_386_JMP_SLOT, R_386_GLOB_DAT, R_386_RELATIVE
5. Summary

TOC: Two syntactic constructs

- Code and data syntactic constructs `.got`, `.plt`
- Global symbols and library function calls
- GOT / PLT addreses
- Assembler format for `.got` and `.plt`
- GNU assembler directives : @got
- GNU assembler directives : @gotoff
- GNU assembler directives : @plt
- GOTs / PLTs of an executable and shared libraries
- Reloc sections

Code and data syntactic constructs .got, .plt

- When the linker creates executables and shared libraries, the linker creates
  - code syntactic constructs (`.plt`)
  - data syntactic constructs (`.got`)
- these were not explicit in the .o files.
- both are *helpers* to the code segment
- since the code segment cannot be modified at run-time

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

Global symbols and library function calls

- a .got section created in the data segment holds pointers to global symbols
  - run time fixups
  - only one entry per application (executable) or
  - only one entry per library
- a .plt section created in the code segment is an array of function stubs used to handle
  - run time resolution of *library calls*.

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

GOT / PLT addresses

- GLOBAL_OFFSET_TABLE : a pointer to the .got
- .got == &GOT[0] : Global Offset Table Address
- .plt == &PLT[0] : Procedure Lookup Table Address

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Assembler format for .got and .plt

- the compiler can signal to the assembler that it wants to trigger .got or .plt constructs by:

implicit func	i386 syntax	ARM syntax
.got pointer	var@GOT(%ebx)	var(GOT)
.got data	var@GOTOFF(%ebx)	var(GOTOFF)
GLOBAL_OFFSET_TABLE	the same	the same
.plt jump	func@PLT	func(PLT)

- Note that the C/C++ programmer does not allocate this memory; it is created by, and used by the linker

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

GNU Assembler directives : @got

- `var@GOT(%ebx)`
- can be used for `.short`, `.long` and `.quad`
- the symbol `var` is added to the GOT
- The symbol term (reference) is replaced with offset from the start of the GOT to the GOT slot for the symbol

`https://web.eecs.umich.edu/~prabal/teaching/resources/eecs373/Assembler.pdf`

GNU Assembler directives : `@gotoff`

- `var@GOTOFF(%ebx)`
- can be used for `.short`, `.long` and `.quad`
- the symbol term (reference) is replaced with the offset from the start of the GOT to the address of the symbol

`https://web.eecs.umich.edu/~prabal/teaching/resources/eecs373/Assembler.pdf`

GNU Assembler directives : @plt

- fun@PLT
- can be used for .long and .quad
- a PLT entry is generated for the function symbol
- the symbol term is replaced with the address of the PLT entry for the symbol.

https://web.eecs.umich.edu/~prabal/teaching/resources/eecs373/Assembler.pdf

_GLOBAL_OFFSET_TABLE_

- A GOT format and interpretation are processor-specific.
- The symbol _GLOBAL_OFFSET_TABLE_ can be used to access the table.
- This symbol can reside in the *middle* of the .got section, allowing both negative and nonnegative subscripts into the array of addresses.
- The symbol type is an array of Elf32_Addr for 32–bit code, and an array of Elf64_Addr for 64–bit code.

```
extern  Elf32_Addr  _GLOBAL_OFFSET_TABLE_[];
extern  Elf64_Addr  _GLOBAL_OFFSET_TABLE_[];
```

https://docs.oracle.com/cd/E23824_01/html/819-0690/chapter6-74186.html

GOTs / PLTs of an executable and shared libraries

- The GOT converts position-independent *address calculations* to absolute locations.
- The PLT converts position-independent *function calls* to absolute locations.

- an executable file has its own GOT and PLT and a shared object file has different GOT and PLT
- an executable and shared object do not share a GOT nor a PLT

https://docs.oracle.com/cd/E23824_01/html/819-0690/chapter6-74186.html

Reoc sections

`.rel.bss`	contains all the `R_386_COPY` relocs
`.rel.plt`	contains all the `R_386_JMP_SLOT` relocs these modify the 1*st* half of the GOT elements
`.rel.got`	contains all the `R_386_GLOB_DATA` relocs these modify the 2*nd* half of the GOT elements
`.rel.data`	contains all the `R_386_32` and `R_386_RELATEIVE` relocs

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

TOC: GOT / PLT relocs in object files

1. R_386_GOT32	a global symbol
2. R_386_GOTOFF	a local symbol
3. R_386_PLT32	a function symbol
4. R_386_GOT	compute &GOT[0]

(1a) R_386_GOT32 : a global symbol

- R_386_GOT32 : reference to a global symbol
- is resolved to the address pointing to the GOT entry for a given global symbol
- can exist in the code area
- persist through the link stage
    - R_386_GOT32 can be seen only in .o files
    - will be converted into R_386_GLOB_DAT at a GOT entry in .so files or executables

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(1b) R_386_GOT32 : a global symbol

- R_386_GOT32 at the global symbol reference
  - distance from GOT[0] (GLOBAL_OFFSET_TABLE) to the GOT entry for a given global symbol
- at the link time, an entry is created in the GOT the GOT entry has a R_386_GLOB_DAT reloc pointing to the global symbol in the library
- at the run time, R_386_GLOB_DAT reloc is filled with the global symbol's address

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(2a) R_386_GOTOFF : a local symbol

- R_386_GOTOFF at a local symbol reference
  in the code section
- a local symbol may be defined in .data ore .bss
- the reloc offset is
  the distance from GOT[0] (GLOBAL_OFFSET_TABLE)
  to a given local symbol

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(2b) R_386_GOTOFF : a local symbol

- R_386_GOTOFF cannot be seen in .so files but only in .o files because it is resolved at the link time
- it cannot exist at a local symbol reference in .data but in .text

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(3a) R_386_PLT32 : a function symbol

- R_386_PLT32 : reference to a function symbol
- is resolved pointing to the PLT entry for a given function symbol
- can exist in the code area
- persist through the link stage
  - R_386_PLT32 can be seen only in .o files
  - will incur R_386_JMP_SLOT in .so files

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(3b) R_386_PLT32 : a function symbol

- R_386_PLT32 at the function symbol reference
  - distance from here (PC-relative)
    to the PLT entry for a given function symbol
- at the link time, an entry is created in the PLT and GOT
  the GOT entry has a R_386_JMP_SLOT reloc
  pointing to the function symbol in the library
- at the run time, the GOT entry is filled with
  the actual symbol values (the function symbol's address)

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

(4a) R_386_GOTPC : compute &GOT[0]

- used in function prolog to calculate &GOT[0]
- R_386_GOTPC determine the distance from here
  to the GLOBAL_OFFSET_TABLE (&GOT[0])
  and deposit the difference as a dword into this location
  (does not involve a symbol!)

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

TOC: Transformed relocs in shared libraries or executable files

1. R_386_GLOB_DAT	a global symbol
2. R_386_RELATIVE	a local symbol
3. R_386_JMP_SLOT	a function symbol

(1a) R_386_GLOB_DAT : a global symbol

- Used to set a GOT entry to
  the address of the specified symbol.
- This special relocation type enable you
  to determine the correspondence
  between symbols and GOT entries

`https://docs.oracle.com/cd/E23824_01/html/819-0690/chapter6-74186.html`

(1b) R_386_GLOB_DAT : a global symbol

- R_386_GLOB_DAT can exist at the 2nd half of GOT entries (.got)
- at dynamic link time, deposit
  the address of a symbol (a subroutine)
  into this dword
- the symbol is in another module
- the complement of the R_386_COPY
  - instead of R_386_GLOB_DAT,
    R_386_COPY could be used.

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(2a) R_386_RELATIVE : a local symbol

- Created by the link-editor for dynamic objects.
- The relocation offset member gives
  the location within a shared object
  that contains a value
  representing a relative address.
- The runtime linker computes the corresponding virtual address
  by adding the virtual address
  at which the shared object is loaded to the relative address.

`https://docs.oracle.com/cd/E23824_01/html/819-0690/chapter6-74186.html`

(2b) R_386_RELATIVE : a local symbol

- at dynamic link time, read the dword at this location, add it to the run-time start address of this module; deposit the result back into this dword
- Relocation entries for this type must specify a value of zero for the symbol table index.

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(3a) R_386_JMP_SLOT : a function symbol

- Created by the link-editor for dynamic objects
  to provide lazy binding
- the relocation offset member gives
  the location of a PLT entry.
- the runtime linker modifies the PLT entry
  to transfer control to the designated symbol address

https://docs.oracle.com/cd/E23824_01/html/819-0690/chapter6-74186.html

(3b) R_386_JMP_SLOT : a function symbol

- R_386_JMP_SLOT can exist at the 1st half of GOT entries (.got.plt)
- at load time, deposit
  the address of a symbol into this dword;

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

TOC: Summary

- Summary- PIC relocs in design cycles
- PIC reloc offsets in an object `.o` file
- PIC reloc offsets in a shared library `.so` file

Summary - PIC relocs in design cycles

	reference in .o	reference in .so
a global symbol	R_386_GOT32	R_386_GLOB_DAT
a local symbol (code)	R_386_GOTOFF	fully resolved
a local symbol (data)	R_386_PC32	R_386_RELATIVE
a function symbol	R_386_PLT32	R_386_JMP_SLOT

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

PIC reloc offsets in an object .o file

R_386_GLOB_DAT $G + A$	• pointing to the GOT entry • distance from GOT[0] to the GOT entry • offset from the start of the GOT to the GOT slot
R_386_GOTOFF $S + A - GOT$	• pointing to the GOT • distance from GOT[0] to the given symbol • offset from the start of the GOT to the symbol
R_386_PC32 $S + A - P$	• pointing to a section (.bss, .data, .text) • distance from a section to the given symbol • offset from the start of a section to the symbol
R_386_PLT32 $L + A - P$	• pointing the PLT entry • distance from the symbol reference to the PLT entry • the address of the PLT entry

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

PIC reloc offsets in a shared library .so file

R_386_GLOB_DAT S	• pointing to the GOT entry • distance from GOT[0] to the GOT entry • offset from the start of the GOT to the GOT slot
R_386_RELATIVE $B + A$	• pointing to a section • distance from a section to the given symbol • offset from the start of a section to the symbol
R_386_JMP_SLOT S	• pointing the PLT entry • distance from the symbol reference to the PLT entry • the address of the PLT entry

https://docs.oracle.com/cd/E19683-01/817-3677/chapter6-26/index.html

TOC: Relocs Summary in i386

1. Background
2. Relocs in .o files for executabls
   R_386_32, R_386_PC32
3. Relocs in .o files for shared libraries
   R_386_GOT32, R_386_GOTOFF, R_386_PLT32, R_386_GOTPC
4. Relocs in executable files
   R_386_COPY, R_386_JMP_SLOT, R_386_GLOB_DAT
5. Relocs in shared library files
   R_386_JMP_SLOT, R_386_GLOB_DAT, R_386_RELATIVE
6. Reloc sections

TOC: 0. Backgorund

- PC-relative offset example
- Reloc legends
- Relocs in PIC object (`.o`) files
- Relocs in PIC shared object (`.so`) fles
- Reloc transformation

PC-relative offset example (1) jump forward

Jump Forward

```
8: 7e 11                jle  1b <silly+0x1b>     Target = dest2
a: 8d b6 00 00 00 00    lea  0x0(%esi),%esi      Added nops
```

- jump target : 0x1b (27)
- jump instruction encoding : 0x7e 0x11
- next instruction address : 0xa (10)
- jump target encoding : 0x1b = 0x11 + 0xa (17 + 10 =27)

Computer Architecture : A Programmer's Prespective

PC-relative offset example (2) jump backward

Jump Backward

```
7.   19: 7f f5                jg    10 <silly+0x10>     Target = dest1
8.   1b: 89 d0                mov   %edx,%eax           dest2:
```

- jump target : 0x10 (16)
- jump instruction encoding : 0x7f 0xf5
- next instruction address : 0x1b (27)
- jump target encoding : 0x10 = 0xf5 + 0x1b (-11 + 27 =16)

Computer Architecture : A Programmer's Prespective

Reloc legends

G	GOT *entry* address from `GOT[0]`
GOT	GOT base address
A	addend
P	current location (symbol reference)
S	symbol address
L	PLT *entry* address

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

Relocs in PIC object (.o) files (1)

R_386_GOT32 for a global symbol reference in the code section
the relative distance of the slot (GOT entry) from GOT[0]
the linker will store a pointer to the given global symbol
used to indirectly reference a global symbol
R_386_GOTOFF for a local symbol reference in the code section
the relative distance of the given symbol from GOT[0]
the linker has placed a pointer to the given local symbol
used to address static data (a local symbol)
R_386_PLT32 for an external function call
the relative distance from the symbol reference to the PLT entry
the linker will store a pointer to the corresponding GOT entry
GOT entry is used to indirectly reference a function symbol

Linkers and Loaders, J. R. Levine

Relocs in PIC object (.o) files (2)

R_386_32 for a global symbol reference in the data section
references the symbol by the name
R_386_32 for a local symbol reference in the data section
references the symbol by the section number (section–offset)
R_386_PC32 for a local function call in the code section
PC-relative calls to a local function

Linkers and Loaders, J. R. Levine

Relocs in PIC shared object (.so) files

R_386_GLOB_DAT for global symbols

used for a global symbol reference in PIC shared libraries

R_386_RELATIVE for local symbols

used to mark data address in a PIC shared library
that need to be relocated at load time

R_386_JMP_SLOT for function symbols

used for a function symbol reference in PIC shared libraries

Linkers and Loaders, J. R. Levine

Reloc transformation

R_386_GOT32	$G + A$	GOT-relative, GOT entry address
R_386_GOTOFF	$S + A - GOT$	symbols in .data, .bss
R_386_32	$S + A$	symbols in .data, .bss, .text
R_386_PLT32	$L + A - P$	PC-relative, PLT entry address

R_386_GOT32	global symbols	R_386_GLOB_DAT
R_386_GOTOFF	local symbols in the code	fully resolved
R_386_32	local symbols in the data	R_386_RELATIVE
R_386_PLT32	function symbols	R_386_JMP_SLOT

R_386_GLOB_DAT	S	fill the global symbol address
R_386_RELATIVE	$B + A$	add the load address for local symbols
R_386_JMP_SLOT	S	fill the function symbol address

Linkers and Loaders, J. R. Levine

TOC: 1. Relocs in .o files for executables

- non-PIC relocs
- R_386_32
- R_386_PC32

non-PIC relocs

R_386_32	(S+A)	for absolute address
R_386_PC32	(S+A-P)	for PC-relative address

(1) R_386_32

- R_386_32 (S+A) absolute address
  - simply *store* the absolute memory address of a symbol at the symbol reference location

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(2) R_386_PC32

- R_386_PC32 (S+A-P) PC-relative address
  - compute the distance
    from the a symbol reference location *to* the symbol,
  - then add it to the current runtime value of the PC
    of the symbol reference instruction
  - store the result at the symbol referece location

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

TOC: 2. Relocs in .o files for shared libraries

- GOT / PLT based relocs
- GOT / PLT based relocs with legends
- R_386_GOT32 in .o files for shared libraries
- R_386_GOTOFF in .o files for shared libraries
- R_386_PLT32 in .o files for shared libraries
- R_386_GOTPC in .o files for shared libraries

GOT / PLT based relocs

- can be seen only in .o files
  which will constitute dynamic libraries (PIC)

R_386_GOT32 (G+A)	GOT-relative, GOT entry address from GOT[0]	global symbols
R_386_GOTOFF (S+A-GOT)	GOT-relative, symbol address from GOT[0]	local symbols
R_386_PLT32 (L+A-P)	PC-relative, PLT entry address from the symbol reference	func symbols
R_386_GOTPC (GOT+A-P)	PC-relative, GOT base address from the current location	func prolog

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

GOT / PLT based relocs with legends

R_386_GOT32	G	GOT *entry* address from GOT[0]
(G+A)		
R_386_GOTOFF	S	symbol address
(S+A-GOT)	GOT	GOT base address
R_386_PLT32	L	PLT *entry* address
(L+A-P)	P	current location (symbol reference)
R_386_GOTPC	GOT	GOT base address
(GOT+A-P)	P	current location

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

(1) R_386_GOT32 in .o files for shared libraries

- R_386_GOT32 (G+A) for a global symbol
  - this reloc is going to persist through the link process
  - this will incur R_386_GLOB_DAT in the library
  - the linker should create this in the GOT entry

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(2) R_386_GOTOFF in .o files for shared libraries

- R_386_GOTOFF (S+A-GOT) for a local symbol in the code section
  - compute the distance from the GOT to the symbol
  - store it at the symbol reference location (resolved)
  - will be fully relsolved at the link time
- R_386_32 (S+A) for a local symbol in the data section
  - references the section number and have a section-offset (.data, .bss, .text)
  - will be changed into a R_386_RELATIVE (B+A) to add the load address to the offset

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(3) R_386_PLT32 in .o files for shared libraries

- R_386_PLT32 (L+A-P) for a function symbol
  - create a new entry in the PLT[ ] and GOT[ ]
  - compute the distance from a symbol reference to the PLT[ ] entry
  - store the computed distance at the symbol reference location
  - the PLT entry points an GOT entry address
  - this reloc will incur R_386_JMP_SLOT
    to fill the GOT[ ] entry with the symbol value (function address)

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(4) R_386_GOTPC in .o files for shared libraries

- R_386_GOTPC (GOT+A-P)
  - compute the difference from here to the GLOBAL_OFFSET_TABLE (&GOT[0])
  - at the definition of each public function which can be called from other modules (does not involve a symbol reference!)
  - used in function prolog to calculate &GOT[0]
  - the function prolog contains something like

    ```
    mov &GOT[0], %ebx
    ```

  - overhead when compiled with -fPIC

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

TOC: 3. Relocs in executable files

- Relocs in static executables
- Relocs in dynamic executables
- Relocs in non-PIC dynamic executable files
- R_386_COPY for non-PIC dynamic executable files
- R_386_JMP_SLOT for non-PIC dynamic executable files

Relocs in static executables

- executable built with static only
  no relocs - run stand alone

`http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html`

Relocs in dynamic executables

- executable with shared libraries (dynamic executables)
  - an executable are usuallly non-PIC
    - the executable does not have its own GOT / PLT
    - `R_386_COPY`, `R_386_JMP_SLOT`
  - nowdays, an executable is PIE by default
    - though not compiled with -fPIC
    - Position Independent Executable
    - the executable has its own GOT / PLT
    - `R_386_JMP_SLOT`, `R_386_GLOB_DAT`, `R_386_RELATIVE`

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

Relocs in non-PIC dynamic executable files

R_386_COPY	• *non-PIC* reference to a global symbol • when a *non-PIC* executable references the global symbol in a shared library • copy the library symbol data into app's data space • offset : a location in a WR segment
R_386_JMP_SLOT	• *non-PIC* reference to a function symbol • when a *non-PIC* executable references the function symbol in a shared library • fill the location with a function symbol address • offset : a PLT entry location of a *PIC* shared library

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(1) R_386_COPY for non-PIC dynamic executable files

- R_386_COPY for intialized data in a library
- read a string of bytes from the symbol address and store a copy into a writable location
- move initialized data from a library down into the application data space (writable)
- offset member : a location in a WR segment (r_offset)
- the "symbol" object has an intrinsic length

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(2) R_386_JMP_SLOT for non-PIC dynamic executable files

- non-PIC executable does not have its own GOT / PLT
- using GOT / PLT of a PIC shared library

- R_386_JMP_SLOT for a function symbol
  - at dynamic link time, the system stores the symbol address into this dword
  - so the corresponding GOT entry will have the target function address
  - this enables indirect jump to procedure through the GOT entry

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(3) Relocs for PIE dynamic executable files

- nowdays, an executable is PIE by default
  - though not compiled with -fPIC
  - a dynamic executable has its own GOT and PLT
- `R_386_JMP_SLOT`, `R_386_GLOB_DAT`, `R_386_RELATIVE` relocs is described in "Relocs in shared libaries"

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

TOC: 4. Relocs in shared library

- Relocs in shared library files
- R_386_JMP_SLOT for shared libary files
- R_386_GLOB_DAT for shared library files
- R_386_RELATIVE for shared library files

Relocs in shared library files

R_386_GLOB_DAT	when a shared library file references the global symbol in other shared library
R_386_JMP_SLOT	when an shared library file references the function symbol in other shared library
R_386_RELATIVE	when a shared library file references the local symbol in the same shared library

R_386_32	can appear in shared library as well.
R_386_PC32	These must be executed carefully.

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(1) R_386_RELATIVE for shared library files

- R_386_RELATIVE
  - at dynamic link time, read the dword at this location
  - add it to the run-time start address of this module
  - store the result back into this dword $(B + A)$

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(2) R_386_JMP_SLOT for shared library files

- R_386_JMP_SLOT for a function symbol
  - at dynamic link time, the system stores the symbol address into this dword
  - so the corresponding GOT entry will have the target function address
  - this enables indirect jump to procedure through the GOT entry

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html

(3) R_386_GLOB_DAT for shared library files

- R_386_GLOB_DAT for a global symbol in other module
  - at load time, store the symbol address into this dword;
  - the "symbol" is in another module - a global symbol
    - this reloc looks like the complement of the R_386_COPY

http://netwinder.osuosl.org/users/p/patb/public_html/elf_relocs.html