GAS Tutorial - 4. Sections & Relocation

Young W. Lim

2016-03-01 Tue

Outline

Based on

"Using as", Dean Elsner, Jay Fenlason & friends

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

Symbols

- the programmer uses symbols to name things
- the linker uses symbols to link
- the debugger uses symbols to debug.

Warning: as does not place symbols in the object file in the same order they were declared.

Labels

- a symbol immediately followed by a colon ':'
- represents the current value of the active location counter
- can be used as instruction operand
- thus, a label should refer only one location
- the first definition overrides any subsequent definitions (warning)

Assign a Value to a Symbol

- writing a symbol, followed by ‘=’, followed by an expression
- .set symbol, expression

- writing a symbol, followed by ‘==’, followed by an expression
- .eqv symbol, expression (snapshot value)

Symbol Names

- symbol names begin with a letter or with one of '.' '_'
- on most machines, you can also use '$' in symbol names
- case sensitive

Local Symbol Names

- begins with a local label prefixes
- the default prefix
  - '.L' for ELF systems
  - 'L' for a.out systems

Local symbols

- defined and used within the assembler
- normally not saved in object files
- invisible in the debugger
- to preserve the local symbols in object files, use '-L' options

Local Labels

- temporary names
- unique symbols in the input source code
- referred to by a simple notation
- possible to repeatedly define the same local label

- to define a local label, write 'N:' (integer N)
- 'Nb' refers to the previous definition of 'N;' (backward N:)
- 'Nf' refers to the next definition of 'N:' (forward N:)

the first 10 local labels ('0:'. . . '9:') are implemented in a slightly more efficient manner than the others.

Local Label Examples

```
Here is an example:
  1:         branch   1f
  2:         branch   1b
  1:         branch   2f
  2:         branch   1b
Which is the equivalent of:
  label_1:   branch   label_3
  label_2:   branch   label_1
  label_3:   branch   label_4
  label_4:   branch   label_3
```

Preserving local labels

- local label are immediately converted into normal symbol names before assembly
- these converted symbols
  - stored in the symbol table
  - appear in error messages
- if '-L' option is used, then the local labels
  - are preserved in the object file
  - may use them in debugging

Local Label Name (1)

an example : .L3C-B44
local label names are constructed as follows:

- local label prefix (.L)
- number (3)
- C-B (Ctrl-B)
- ordinal number (44)

Local Label Name (2)

1. Local Label Prefix
1. Number : the number that was used in N:
1. C-B : The special character of '02' (control-B) for differentiation
1. Ordinal Number
   - a serial number to keep the labels distinct.
   - 1 : for the first definition of '0:'
   - 16 : for the 15th definition of '0:'

- the first 1: may be named .L1C-B1
- the 44th 3: may be named .L3C-B44

Dollar Local Labels

- ‘N$:’
- converted symbol name uses the ‘01’ (control-A)
- local (valid for only a part of the input source code)
- out of scope (undefined) as soon as a non-local label is defined
- the 5-th definition of ‘6$’ may be named ‘.L6C-A5’.

Dot Symbol

- The special symbol '.' refers to the current address
- 'melvin: .long .' defines melvin to contain its own address
- assigning a value to . is treated the same as a .org directive.
- the expression '.=.+4' is the same as saying '.space 4'.

Symbol Attribute

Every symbol has

- name
- the attribute “value”
- the attribute “type”
- auxiliary attributes

- the attribute of undefined symbols : zero
- externally defined symbols

Symbol Attribute - Value (1)

- usually 32 bits.
- For a symbol which labels a location
  - the value is the number of addresses
  - from the start of that section to that label

Symbol Attribute - Value (2)

- the value of a symbol in text, data, and bss section changes as ld changes section base addresses during linking.
- absolute symbols'values do not change during linking
- undefined symbol if its value is 0
  - ld tries to determine its value from other files
- by mentioning a symbol name without defining
- by using .comm common declaration.
  - the value is how much common storage to reserve, in bytes (addresses)
  - the symbol refers to the first address of the allocated storage.

Symbol Attribute - Type

The type attribute of a symbol contains

- relocation (section) information
- any flag settings indicating that a symbol is external
- other information for linkers and debuggers.

TTTT