GAS Tutorial - 6. Expression

Young W. Lim

2016-03-03 Wed

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.

Expression

- An expression : either address or numeric value
  - an absolute number
  - an offset into a particular section.
- if an expression is not absolute
- and if not enough information then error

Empty Expression

- An empty expression has no value
    - just whitespace
    - null
    - 0 if an absolute expression is needed

Integer Expression

- An integer expression
  - one or more arguments delimited by operators
- Arguments
  - Arguments are symbols, numbers or subexpressions.
- Operators
  - arithmetic functions, like or %.

Arguments

- symbols, numbers or subexpressions
- arithmetic operands
- Symbols are evaluated to yield {section NNN }
  - section is one of text, data, bss, absolute, or undefined
  - NNN is a signed, 2's complement 32 bit integer
- Numbers are usually integers
- A number can be a flonum or bignum
- warned that only the low order 32 bits are used
- integer-manipulating instructions
- Subexpressions are
  - '(' an integer expression ')';
  - a prefix operator followed by an argument.

Operators

- Operators are arithmetic functions, like + or %.
- Prefix operators are followed by an argument.
- Infix operators appear between their arguments.
- Operators may be preceded and/or followed by whitespace.

Infix Operators

- Infix operators take two arguments, one on either side.
- Operators have precedence, but operations with equal

precedence are performed left to right.

- Apart from + or '-', both arguments must be absolute,

and the result is absolute.

Infix Operators (1)

```
1. Highest Precedence
    *          Multiplication.
    /          Integer Division.
    %          Remainder.
    <<         Shift Left. Same as the C operator ‘<<’.
    >>         Shift Right. Same as the C operator ‘>>’.
2. Intermediate precedence
    |
               Bitwise Inclusive Or.
    &          Bitwise And.
    ^          Bitwise Exclusive Or.
    !          Bitwise Or Not.
```

Infix Operators (2)

```
3. Low Precedence
    +          Addition. If either argument is absolute,
               the result has the section of the other
               argument. You may not add together arguments
               from different sections.
    -          Subtraction. If the right argument is
               absolute, the result has the section
               of the left argument. If both arguments
               are in the same section, the result is
               absolute. You may not subtract arguments
               from different sections.
```

Infix Operators (3)

`==`	Is Equal To
`<>`	
`!=`	Is Not Equal To
`<`	Is Less Than
`>`	Is Greater Than
`>=`	Is Greater Than Or Equal To
`<=`	Is Less Than Or Equal To

```
The comparison operators can be used as
infix operators. A true results has a value
of -1 whereas a false result has a value
of 0. Note, these operators
perform signed comparisons.
```

Infix Operators (4)

```
4. Lowest Precedence
    &&        Logical And.
    ||        Logical Or.
              These two logical operations can be used
              to combine the results of sub expressions.
              Note, unlike the comparison operators a true
              result returns a value of 1 but a false
              results does still return 0. Also note that
              the logical or operator has a slightly
              lower precedence than logical and.
```

Symbols

In short, it's only meaningful to add or subtract the offsets in an address; you can only have a defined section in one of the two arguments.

TTTT