The Go Tool Chain

Ben Shi

Brief Introduction (1)

- Born in 2009 in Google by the UNIX/C team
- Keep it simple and stupid
- Ranking 17 in TIOBE Index in September 2017
- Usage: clouds server / web / container

Brief Introduction (2)

- Static language, strongly typed
- Garbage collection
- Multiple return values
- Simple error handling
- Built in array, slice and map
- Anonymous function
- Goroutine
- Channel
- OOP

Multiple Return Values

apuser@tj07598pcu: ~/Desktop

```c
#include <stdio.h>

struct res {
        int sum;
        int dif;
};

void add_sub(struct res *pr, int a, int b)
{
        pr->sum = a + b;
        pr->dif = a - b;
}

int main()
{
        struct res r;
        add_sub(&r, 12, 2);
        printf("%d\t%d\n", r.sum, r.dif);
        return 0;
}
~
~
a.c                    1,1           All
```

```go
package main
import "fmt"

func main() {
        i, j := add_sub(12, 2)
        fmt.Println(i, j)
}

func add_sub(a int, b int) (int, int) {
        return a + b, a - b
}
~
~
~
~
~
~
~
~
~
~
a.go                   1,1           All
```

Error Handling

```java
// TODO Auto-generated method stub
// file(内存)----输入流---->【程序】----输出流---->file(内存)
File file = new File("d:/temp", "addfile.txt");
try {
    file.createNewFile(); // 创建文件
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// 向文件写入内容(输出流)
String str = "亲爱的小南瓜!";
byte bt[] = new byte[1024];
bt = str.getBytes();
try {
    FileOutputStream in = new FileOutputStream(file);
    try {
        in.write(bt, 0, bt.length);
        in.close();
        // boolean success=true;
        // System.out.println("写入文件成功");
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

apuser@tj07598pcu: ~/Desktop

```go
package main

import (
        "log"
        "os"
        "fmt"
)

func main() {
        f, err := os.OpenFile("notes.txt", os.O_CREATE, 0755)
        if err != nil {
                log.Fatal(err)
        }
        defer f.Close()
        b := make([]byte, 200)
        if _, err := f.Read(b); err != nil {
                log.Fatal(err)
        }
        fmt.Println(b)
}
~
~
~
"a.go" 20L, 269C
```

Builtin Array, Slice & Map

```go
package main

import (
        "fmt"
)

func main() {
        var a [10]int
        var b []int = make([]int, 10)
        var c map[string]int = make(map[string]int, 10)

        a[0] = 1
        b[2] = 2
        c["Three"] = 3
        fmt.Println(a, b, c)

        b = a[1:3]
        c["Four"] = 4
        fmt.Println(a, b, c)
}
```

```
apuser@tj07598pcu:~/Desktop$ go run a.go
[1 0 0 0 0 0 0 0 0 0] [0 0 2 0 0 0 0 0 0 0] map[Three:3]
[1 0 0 0 0 0 0 0 0 0] [0 0] map[Three:3 Four:4]
apuser@tj07598pcu:~/Desktop$
```

Array VS Slice

```go
package main
import "fmt"
func f0(slice []int32) int32 {
        slice[0]++
        return slice[0]
}
func f1(array [10]int32) int32 {
        array[0]++
        return array[0]
}
func main() {
        slice := make([]int32, 10)
        slice[0] = 34
        array := [10]int32{12, 3, 3, 4, 5, 65, 7, 8, 9, 0}
        fmt.Printf("%d\t%d\n", f0(slice), f1(array))
        fmt.Printf("%d\t%d\n", slice[0], array[0])
}
```

```
apuser@tj07598pcu:~$ go run a.go
35      13
35      12
```

Anonymous Function

apuser@tj07598pcu: ~

```c
#include <stdio.h>

int f0(int b)
{
        return b + 2;
}

int f1(int b)
{
        return b + 100;
}

int main()
{
        printf("%d\t%d\n", f0(10), f1(10));
        return 0;
}
~
~
~
~
~
```

b.c 1,1 All

```go
package main

import "fmt"

func fff(a int) (func (int) int) {
        return func (b int) int {
                // static int a
                return a + b
        }
}

func main() {
        f0 := fff(2)
        f1 := fff(100)
        fmt.Println(f0(10), f1(10))
}
~
~
~
~
~
~
```

a.go 1,1 All

For Loop

```go
package main
import "fmt"
func main() {
        a := map[string]string{}
        a["red"] = "红"
        a["green"] = "绿"
        a["blue"] = "蓝"
        for i, v := range a {
                fmt.Println(i, v)
        }
}
```

```
apuser@tj07598pcu:~$ go run a.go
red 红
green 绿
blue 蓝
```

Goroutine

apuser@tj07598pcu: ~/Desktop

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h>

struct param {
        const char msg[16];
        int n;
};

void *rt(void *p)
{
        struct param *pp = (struct param *) p;
        for (int i = 0; i < pp->n; i++) {
                usleep(1000 * (500 + (random() % 500)));
                printf("%s\n", pp->msg);
        }
        return NULL;
}

int main()
{
        struct param a = {"AAAAAA", 10};
        struct param b = {"BBBBBB", 10};
        pthread_t pa, pb;
        pthread_create(&pa, NULL, rt, &a);
        pthread_create(&pb, NULL, rt, &b);
        for (;;) {}
        return 0;
}
~
```

b.c 1,1 All

```go
package main

import (
        "fmt"
        "math/rand"
        "time"
)

func rt(c string, n int) {
        for i := 0; i < n; i++ {
                t := 500 + time.Duration(rand.Intn(500))
                time.Sleep(t * time.Millisecond)
                fmt.Println(c)
        }
}

func main() {
        go rt("AAAAAAAAAAAAA", 10)
        go rt("BBBBBBBBBBBBB", 10)
        for ;; {}
}
~
~
~
~
~
~
~
~
~
~
```

a.go 1,1 All

Channel

apuser@tj07598pcu: ~/Desktop

apuser@tj07598pcu: ~/Desktop | apuser@tj07598pcu: ~/Deskto... | apuser@tj07598pcu: ~/Desktop

```c
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <fcntl.h>
#include <stdlib.h>
int main()
{
        pid_t pid;
        int f[2];

        if (pipe(f) != 0) {
                perror(NULL);
                return EXIT_FAILURE;
        }

        if ((pid = fork()) == 0) {
                for (int i = 100; i < 110; i++)
                        write(f[1], &i, sizeof(i));
        } else if (pid > 0) {
                int j;
                for (int i = 0; i < 10; i++) {
                        read(f[0], &j, sizeof(j));
                        printf("%d\n", j);
                }
                waitpid(pid, &j, 0);
        }

        return EXIT_SUCCESS;
}
```

```go
package main

import "fmt"

var ch chan int

func send(i, j int) {
        for a := i; a < j; a++ {
                ch <- a
        }
}

func recv() {
        for {
                a := <-ch
                fmt.Println(a)
        }
}

func main() {
        ch = make(chan int)
        go send(0, 20)
        go recv()
        for {
        }
}
~
~
~
~
```

a.c [+] 31,1 Bot | a.go 1,1 All

OOP (1)

apuser@tj07598pcu: ~/Desktop

```go
package main

import "fmt"

type base struct {
        a int
        d int
}

type ext struct {
        base
        a int
        c base
}

func main() {
        var e ext = ext{base{2, 3}, 3, base{4, 6}}
        fmt.Println(e.a, e.base.a, e.c.a, e.d)
}
~
~
~
~
~
~
~
~
~
~
a.go                    2,0-1         All
```

```cpp
#include<iostream>

using namespace std;

class base {
public:
        int a;
        int d;
        base(int b): a(b), d(0) {}
};

class ext:public base {
public:
        int a;
        base c;
        ext(int q, int w, int e): base(q), a(e), c(w) {}
};

int main()
{
        ext e(1, 2, 3);
        cout << e.a << "\t" << e.base::a << "\t"
        << e.c.a << "\t" << e.d << endl;
        return 0;
}
~
~
~
~
c.cc                    1,1           All
```

OOP (2)

apuser@tj07598pcu: ~/Desktop

```go
package main
import "fmt"
type reader interface {
        read(b []byte)
}
type cl0 struct {
        br []byte
}
type cl1 struct {
        br []byte
}
func (r cl0) read(b []byte) {
        copy(b, r.br[1:])
}
func (r cl1) read(b []byte) {
        copy(b, r.br)
}
func main() {
        var c0 reader = cl0{[]byte{65, 66, 67}}
        var c1 reader = cl1{[]byte{97, 98, 99}}
        b0 := make([]byte, 10)
        b1 := make([]byte, 10)
        c0.read(b0)
        c1.read(b1)
        c0.read(b0)
        c1.read(b1)
        fmt.Println(b0, b1)
}
~
```

a.go 28,1 All

```java
interface reader {
        void read(byte[] r);
}
class cl0 implements reader {
        public void read(byte[] r) {
                byte[] a = {65, 66, 67};
                System.arraycopy(a, 0, r, 0, 3);
        }
}
class cl1 implements reader {
        public void read(byte[] r) {
                byte[] a = {97, 98, 99};
                System.arraycopy(a, 0, r, 0, 3);
        }
}
public class f {
        public static void main(String[] args) {
                reader r0 = new cl0();
                reader r1 = new cl1();
                byte[] b0 = new byte[3];
                byte[] b1 = new byte[3];
                r0.read(b0);
                r1.read(b1);
                System.out.println(new String(b0));
                System.out.println(new String(b1));
        }
}
~
~
```

f.java 1,1 All

Golang Builtin Compiler

- Bootstrap from go-1.5
- SSA from go-1.7
- Arch: arm/arm64/x86/amd64/mips/mips64/ppc64
- OS: Linux/Windows/*BSD/Plan9/MacOS
- Less optimized than GCC/LLVM
- In-compatible with GCC

Source -> AST

```
func calc(a, b, c, d, e int) int {
        return (a + b) / (d + e * c)
}
```

```
generating SSA for calc
buildssa-enter
.   AS l(5)
.   .   NAME-main.~r5 a(true) g(1) l(5) x(20) class(PPARAMOUT) f(1) int
buildssa-body
.   RETURN l(6) tc(1)
.   RETURN-list
.   .   AS l(6) tc(1) hascall
.   .   .   NAME-main.~r5 a(true) g(1) l(5) x(20) class(PPARAMOUT) f(1) int
.   .   .   DIV l(6) tc(1) hascall int
.   .   .   .   ADD l(6) tc(1) int
.   .   .   .   .   NAME-main.a a(true) g(2) l(5) x(0) class(PPARAM) f(1) tc(1) used int
.   .   .   .   .   NAME-main.b a(true) g(3) l(5) x(4) class(PPARAM) f(1) tc(1) used int
.   .   .   .   ADD l(6) tc(1) int
.   .   .   .   .   NAME-main.d a(true) g(5) l(5) x(12) class(PPARAM) f(1) tc(1) used int
.   .   .   .   .   MUL l(6) tc(1) int
.   .   .   .   .   .   NAME-main.e a(true) g(6) l(5) x(16) class(PPARAM) f(1) tc(1) used int
.   .   .   .   .   .   NAME-main.c a(true) g(4) l(5) x(8) class(PPARAM) f(1) tc(1) used int
buildssa-exit
```

IR Transforms

Mozilla Firefox

file:///home/apuser/Desktop/ssa.html

calc

help

start

```
b1:

  v1 = InitMem <mem>
  v2 = SP <uintptr>
  v3 = SB <uintptr>
  v4 = Addr <*int> {a} v2
  v5 = Addr <*int> {b} v2
  v6 = Addr <*int> {c} v2
  v7 = Addr <*int> {d} v2
  v8 = Addr <*int> {e} v2
  v9 = Addr <*int> {~r5} v2
  v10 = Arg <int> {a}
  v11 = Arg <int> {b}
  v12 = Arg <int> {c}
  v13 = Arg <int> {d}
  v14 = Arg <int> {e}
  v15 = Const32 <int> [0]
  v16 = Add32 <int> v10 v11
  v17 = Mul32 <int> v14 v12
  v18 = Add32 <int> v13 v17
  v19 = Neq32 <bool> v18 v15
If v19 → b2 b3 (likely)

b2: ← b1

  v22 = Div32 <int> v16 v18
  v23 = Copy <mem> v1
  v24 = VarDef <mem> {~r5} v23
  v25 = Store <mem> {int} v9 v22 v24
Ret v25

b3: ← b1

  v20 = Copy <mem> v1
  v21 = StaticCall <mem> {runtime.panicdivide} v20
Exit v21
```

after early phielim [1666 ns]

```
b1:

  v1 = InitMem <mem>
  v2 = SP <uintptr>
  v3 = SB <uintptr>
  v4 = Addr <*int> {a} v2
  v5 = Addr <*int> {b} v2
  v6 = Addr <*int> {c} v2
  v7 = Addr <*int> {d} v2
  v8 = Addr <*int> {e} v2
  v9 = Addr <*int> {~r5} v2
  v10 = Arg <int> {a}
  v11 = Arg <int> {b}
  v12 = Arg <int> {c}
  v13 = Arg <int> {d}
  v14 = Arg <int> {e}
  v15 = Const32 <int> [0]
  v16 = Add32 <int> v10 v11
  v17 = Mul32 <int> v14 v12
  v18 = Add32 <int> v13 v17
  v19 = Neq32 <bool> v18 v15
If v19 → b2 b3 (likely)

b2: ← b1

  v22 = Div32 <int> v16 v18
  v23 = Copy <mem> v1
  v24 = VarDef <mem> {~r5} v1
  v25 = Store <mem> {int} v9 v22 v24
Ret v25

b3: ← b1

  v20 = Copy <mem> v1
  v21 = StaticCall <mem> {runtime.panicdivide} v1
Exit v21
```

after early copyelim [1115 ns]

```
b1:

  v1 = InitMem <mem>
  v2 = SP <uintptr>
  v3 = SB <uintptr>
  v4 = Addr <*int> {a} v2
  v5 = Addr <*int> {b} v2
  v6 = Addr <*int> {c} v2
  v7 = Addr <*int> {d} v2
  v8 = Addr <*int> {e} v2
  v9 = Addr <*int> {~r5} v2
  v10 = Arg <int> {a}
  v11 = Arg <int> {b}
  v12 = Arg <int> {c}
  v13 = Arg <int> {d}
  v14 = Arg <int> {e}
  v15 = Const32 <int> [0]
  v16 = Add32 <int> v10 v11
  v17 = Mul32 <int> v14 v12
  v18 = Add32 <int> v13 v17
  v19 = Neq32 <bool> v18 v15
If v19 → b2 b3 (likely)

b2: ← b1

  v22 = Div32 <int> v16 v18
  v23 = Copy <mem> v1
  v24 = VarDef <mem> {~r5} v1
  v25 = Store <mem> {int} v9 v22 v24
Ret v25

b3: ← b1

  v20 = Copy <mem> v1
  v21 = StaticCall <mem> {runtime.panicdivide} v1
Exit v21
```

after early deadcode [9620 ns]

```
b1:

  v1 = InitMem <mem>
  v2 = SP <uintptr>
  v9 = Addr <*int> {~r5} v2
  v10 = Arg <int> {a}
  v11 = Arg <int> {b}
  v12 = Arg <int> {c}
  v13 = Arg <int> {d}
  v14 = Arg <int> {e}
  v15 = Const32 <int> [0]
  v16 = Add32 <int> v10 v11
  v17 = Mul32 <int> v14 v12
  v18 = Add32 <int> v13 v17
  v19 = Neq32 <bool> v18 v15
If v19 → b2 b3 (likely)

b2: ← b1

  v22 = Div32 <int> v16 v18
  v24 = VarDef <mem> {~r5} v1
  v25 = Store <mem> {int} v9 v22 v24
Ret v25

b3: ← b1

  v21 = StaticCall <mem> {runtime.panicdivide} v1
Exit v21
```

after short circuit [4635 ns]

```
b1:

  v1 = InitMem <mem>
  v2 = SP <uintptr>
  v9 = Addr <*int> {~r5} v2
  v10 = Arg <int> {a}
  v11 = Arg <int> {b}
  v12 = Arg <int> {c}
  v13 = Arg <int> {d}
  v14 = Arg <int> {e}
  v15 = Const32 <int> [0]
  v16 = Add32 <int> v10 v11
  v17 = Mul32 <int> v14 v12
  v18 = Add32 <int> v13 v17
  v19 = Neq32 <bool> v18 v15
If v19 → b2 b3 (likely)

b2: ← b1

  v22 = Div32 <int> v16 v18
  v24 = VarDef <mem> {~r5} v1
  v25 = Store <mem> {int} v9 v22 v24
Ret v25

b3: ← b1

  v21 = StaticCall <mem> {runtime.panicdivide} v1
Exit v21
```

IR -> Assembly

Mozilla Firefox

Ben Shi (史斌) - Outloc | 球迷沙龙 - 北方论坛 - P | file:///home/apuser/Desk

file:///home/apuser/Desktop/ssa.html

after regalloc [99117 ns]

```
b1:

  v1 = InitMem <mem>
  v10 = Arg <int> {a} : a[int]
  v11 = Arg <int> {b} : b[int]
  v12 = Arg <int> {c} : c[int]
  v13 = Arg <int> {d} : d[int]
  v14 = Arg <int> {e} : e[int]
  v34 = LoadReg <int> v11 : R2
  v33 = LoadReg <int> v10 : R3
  v16 = ADD <int> v34 v33 : R2
  v31 = LoadReg <int> v14 : R3
  v30 = LoadReg <int> v12 : R4
  v29 = LoadReg <int> v13 : R5
  v18 = MULA <int> v31 v30 v29 : R3
  v23 = CMPconst <flags> [0] v18
NE v23 → b2 b3 (likely)

b2: ← b1

  v9 = StoreReg <int> v16 : .autotmp_6[int]
  v26 = StoreReg <int> v18 : .autotmp_7[int]
  v4 = XORshiftRA <uint32> [31] v16 v16 : R4
  v5 = SUBshiftRA <uint32> [31] v4 v16 : R1
  v28 = XORshiftRA <uint32> [31] v18 v18 : R4
  v27 = SUBshiftRA <uint32> [31] v28 v18 : R0
  v6 = CALLudiv <uint32,uint32> v5 v27 : <R0,R1>
  v7 = Select0 <uint32> v6 : R0
  v24 = VarDef <mem> {~r5} v1
  v2 = SP <uintptr> : SP
  v3 = LoadReg <int> v26 : R2
  v20 = LoadReg <int> v9 : R3
  v32 = XOR <uint32> v3 v20 : R2
  v8 = XORshiftRA <uint32> [31] v7 v32 : R3
  v22 = SUBshiftRA <int> [31] v8 v32 : R2
  v25 = MOVWstore <mem> {~r5} v2 v22 v24
Ret v25

b3: ← b1

  v21 = CALLstatic <mem> {runtime.panicdivide} v1
Exit v21
```

after loop rotate [392 ns]

```
b1:

  v1 = InitMem <mem>
  v10 = Arg <int> {a} : a[int]
  v11 = Arg <int> {b} : b[int]
  v12 = Arg <int> {c} : c[int]
  v13 = Arg <int> {d} : d[int]
  v14 = Arg <int> {e} : e[int]
  v34 = LoadReg <int> v11 : R2
  v33 = LoadReg <int> v10 : R3
  v16 = ADD <int> v34 v33 : R2
  v31 = LoadReg <int> v14 : R3
  v30 = LoadReg <int> v12 : R4
  v29 = LoadReg <int> v13 : R5
  v18 = MULA <int> v31 v30 v29 : R3
  v23 = CMPconst <flags> [0] v18
NE v23 → b2 b3 (likely)

b2: ← b1

  v9 = StoreReg <int> v16 : .autotmp_6[int]
  v26 = StoreReg <int> v18 : .autotmp_7[int]
  v4 = XORshiftRA <uint32> [31] v16 v16 : R4
  v5 = SUBshiftRA <uint32> [31] v4 v16 : R1
  v28 = XORshiftRA <uint32> [31] v18 v18 : R4
  v27 = SUBshiftRA <uint32> [31] v28 v18 : R0
  v6 = CALLudiv <uint32,uint32> v5 v27 : <R0,R1>
  v7 = Select0 <uint32> v6 : R0
  v24 = VarDef <mem> {~r5} v1
  v2 = SP <uintptr> : SP
  v3 = LoadReg <int> v26 : R2
  v20 = LoadReg <int> v9 : R3
  v32 = XOR <uint32> v3 v20 : R2
  v8 = XORshiftRA <uint32> [31] v7 v32 : R3
  v22 = SUBshiftRA <int> [31] v8 v32 : R2
  v25 = MOVWstore <mem> {~r5} v2 v22 v24
Ret v25

b3: ← b1

  v21 = CALLstatic <mem> {runtime.panicdivide} v1
Exit v21
```

after stackframe [3462 ns]

```
b1:

  v1 = InitMem <mem>
  v10 = Arg <int> {a} : a[int]
  v11 = Arg <int> {b} : b[int]
  v12 = Arg <int> {c} : c[int]
  v13 = Arg <int> {d} : d[int]
  v14 = Arg <int> {e} : e[int]
  v34 = LoadReg <int> v11 : R2
  v33 = LoadReg <int> v10 : R3
  v16 = ADD <int> v34 v33 : R2
  v31 = LoadReg <int> v14 : R3
  v30 = LoadReg <int> v12 : R4
  v29 = LoadReg <int> v13 : R5
  v18 = MULA <int> v31 v30 v29 : R3
  v23 = CMPconst <flags> [0] v18
NE v23 → b2 b3 (likely)

b2: ← b1

  v9 = StoreReg <int> v16 : .autotmp_6[int]
  v26 = StoreReg <int> v18 : .autotmp_7[int]
  v4 = XORshiftRA <uint32> [31] v16 v16 : R4
  v5 = SUBshiftRA <uint32> [31] v4 v16 : R1
  v28 = XORshiftRA <uint32> [31] v18 v18 : R4
  v27 = SUBshiftRA <uint32> [31] v28 v18 : R0
  v6 = CALLudiv <uint32,uint32> v5 v27 : <R0,R1>
  v7 = Select0 <uint32> v6 : R0
  v24 = VarDef <mem> {~r5} v1
  v2 = SP <uintptr> : SP
  v3 = LoadReg <int> v26 : R2
  v20 = LoadReg <int> v9 : R3
  v32 = XOR <uint32> v3 v20 : R2
  v8 = XORshiftRA <uint32> [31] v7 v32 : R3
  v22 = SUBshiftRA <int> [31] v8 v32 : R2
  v25 = MOVWstore <mem> {~r5} v2 v22 v24
Ret v25

b3: ← b1

  v21 = CALLstatic <mem> {runtime.panicdivide} v1
Exit v21
```

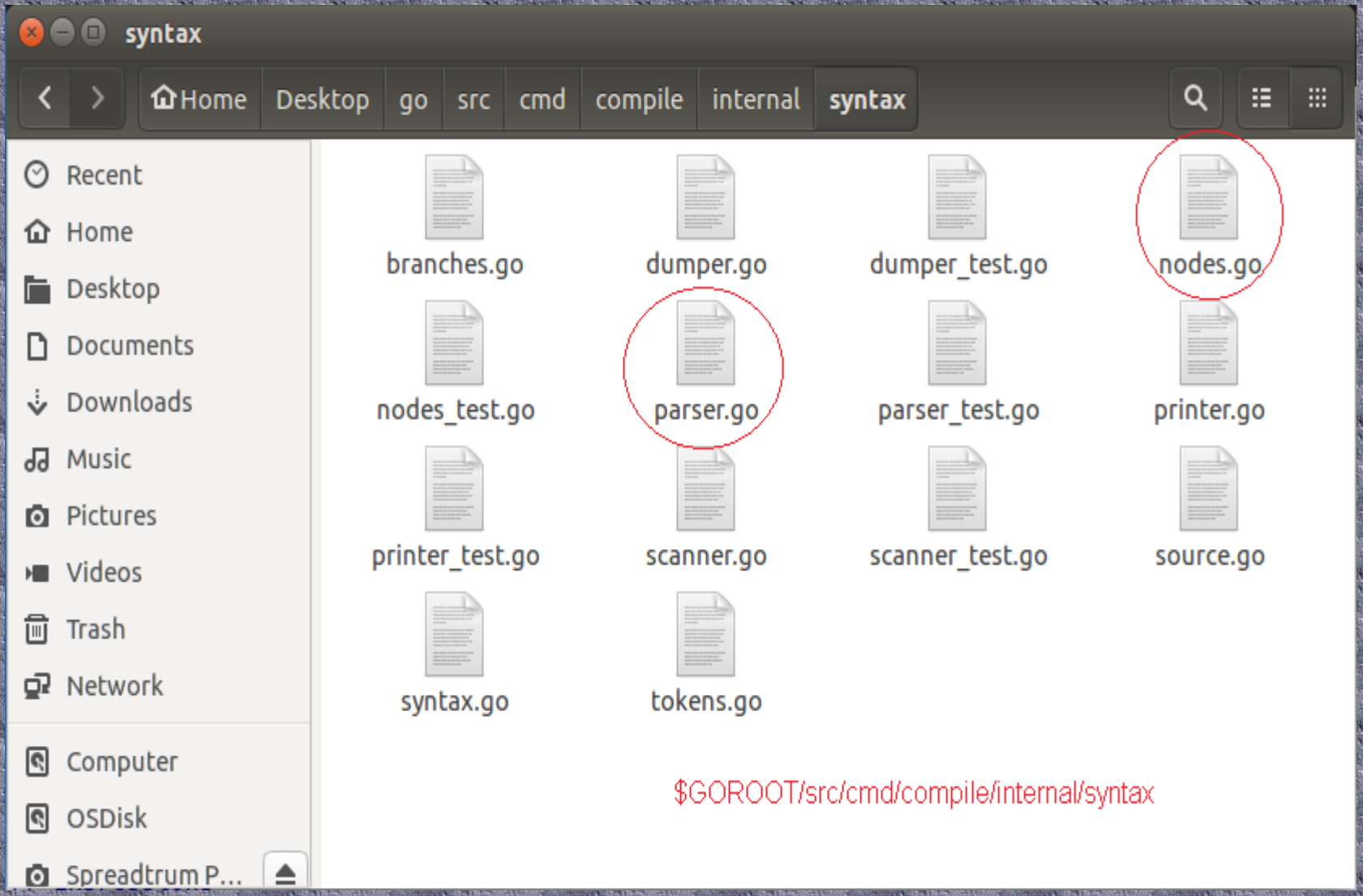
after trim [580 ns]

```
b1:

  v1 = InitMem <mem>
  v10 = Arg <int> {a} : a[int]
  v11 = Arg <int> {b} : b[int]
  v12 = Arg <int> {c} : c[int]
  v13 = Arg <int> {d} : d[int]
  v14 = Arg <int> {e} : e[int]
  v34 = LoadReg <int> v11 : R2
  v33 = LoadReg <int> v10 : R3
  v16 = ADD <int> v34 v33 : R2
  v31 = LoadReg <int> v14 : R3
  v30 = LoadReg <int> v12 : R4
  v29 = LoadReg <int> v13 : R5
  v18 = MULA <int> v31 v30 v29 : R3
  v23 = CMPconst <flags> [0] v18
NE v23 → b2 b3 (likely)

b2: ← b1

  v9 = StoreReg <int> v16 : .autotmp_6[int]
  v26 = StoreReg <int> v18 : .autotmp_7[int]
  v4 = XORshiftRA <uint32> [31] v16 v16 : R4
  v5 = SUBshiftRA <uint32> [31] v4 v16 : R1
  v28 = XORshiftRA <uint32> [31] v18 v18 : R4
  v27 = SUBshiftRA <uint32> [31] v28 v18 : R0
  v6 = CALLudiv <uint32,uint32> v5 v27 : <R0,R1>
  v7 = Select0 <uint32> v6 : R0
  v24 = VarDef <mem> {~r5} v1
  v2 = SP <uintptr> : SP
  v3 = LoadReg <int> v26 : R2
  v20 = LoadReg <int> v9 : R3
  v32 = XOR <uint32> v3 v20 : R2
  v8 = XORshiftRA <uint32> [31] v7 v32 : R3
  v22 = SUBshiftRA <int> [31] v8 v32 : R2
  v25 = MOVWstore <mem> {~r5} v2 v22 v24
Ret v25

b3: ← b1

  v21 = CALLstatic <mem> {runtime.panicdivide} v1
Exit v21
```

genssa

```
      00000 (/home/apuser/Desktop/a.go:5) TEXT
      "".calc(SB)
      00001 (/home/apuser/Desktop/a.go:5) FUNCDATA
      $0,
      gclocals·26c19b003b4032a46d3e8db29831f3fe(SB)
      00002 (/home/apuser/Desktop/a.go:5) FUNCDATA
      $1,
      gclocals·33cdeccccebe80329f1fdbee7f5874cb(SB)
v34   00003 (/home/apuser/Desktop/a.go:5) MOVW
      "".b+4(R13), R2
v33   00004 (/home/apuser/Desktop/a.go:5) MOVW
      "".a(R13), R3
v16   00005 (/home/apuser/Desktop/a.go:6) ADD R3,
      R2, R2
v31   00006 (/home/apuser/Desktop/a.go:6) MOVW
      "".e+16(R13), R3
v30   00007 (/home/apuser/Desktop/a.go:6) MOVW
      "".c+8(R13), R4
v29   00008 (/home/apuser/Desktop/a.go:6) MOVW
      "".d+12(R13), R5
v18   00009 (/home/apuser/Desktop/a.go:6) MULA R3,
      R4, R5, R3
v23   00010 (/home/apuser/Desktop/a.go:6) CMP $0,
      R3
b1    00011 (/home/apuser/Desktop/a.go:6) BEQ 26
v9    00012 (/home/apuser/Desktop/a.go:6) MOVW R2,
      "".autotmp_6-4(R13)
v26   00013 (/home/apuser/Desktop/a.go:6) MOVW R3,
      "".autotmp_7-8(R13)
v4    00014 (/home/apuser/Desktop/a.go:6) EOR
      R2->31, R2, R4
v5    00015 (/home/apuser/Desktop/a.go:6) SUB
      R2->31, R4, R1
v28   00016 (/home/apuser/Desktop/a.go:6) EOR
      R3->31, R3, R4
v27   00017 (/home/apuser/Desktop/a.go:6) SUB
      R3->31, R4, R0
v6    00018 (/home/apuser/Desktop/a.go:6) CALL
      runtime.udiv(SB)
v3    00019 (/home/apuser/Desktop/a.go:6) MOVW
      "".autotmp_7-8(R13), R2
v20   00020 (/home/apuser/Desktop/a.go:6) MOVW
      "".autotmp_6-4(R13), R3
v32   00021 (/home/apuser/Desktop/a.go:6) EOR R3,
      R2, R2
v8    00022 (/home/apuser/Desktop/a.go:6) EOR
      R2->31, R0, R3
v22   00023 (/home/apuser/Desktop/a.go:6) SUB
      R2->31, R3, R2
v25   00024 (/home/apuser/Desktop/a.go:6) MOVW R2,
      "".~r5+20(R13)
b2    00025 (/home/apuser/Desktop/a.go:6) RET
v21   00026 (/home/apuser/Desktop/a.go:6) PCDATA
      $0, $0
v21   00027 (/home/apuser/Desktop/a.go:6) CALL
      runtime.panicdivide(SB)
b3    00028 (/home/apuser/Desktop/a.go:6) UNDEF
      00029 (<unknown line number>) END
```

Parser


syntax
Home
Desktop
go
src
cmd
compile
internal
syntax
Recent
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Network
Computer
OSDisk
Spreadtrum P...
branches.go
dumper.go
dumper_test.go
nodes.go
nodes_test.go
parser.go
parser_test.go
printer.go
printer_test.go
scanner.go
scanner_test.go
source.go
syntax.go
tokens.go
$GOROOT/src/cmd/compile/internal/syntax

Declaration

```
//              Path
// LocalPkgName Path
ImportDecl struct {
        LocalPkgName *Name // including "."; nil means no rename present
        Path         *BasicLit
        Group        *Group // nil means not part of a group
        decl
}

// NameList
// NameList      = Values
// NameList Type = Values
ConstDecl struct {
        NameList []*Name
        Type     Expr   // nil means no type
        Values   Expr   // nil means no values
        Group    *Group // nil means not part of a group
        decl
}

// Name Type
TypeDecl struct {
        Name   *Name
        Alias  bool
        Type   Expr
        Group  *Group // nil means not part of a group
        Pragma Pragma
        decl
}

// NameList Type
// NameList Type = Values
// NameList      = Values
VarDecl struct {
        NameList []*Name
        Type     Expr   // nil means no type
        Values   Expr   // nil means no values
        Group    *Group // nil means not part of a group
        decl
}

// func          Name Type { Body }
// func          Name Type
// func Receiver Name Type { Body }
// func Receiver Name Type
FuncDecl struct {
        Attr   map[string]bool // go:attr map
        Recv   *Field          // nil means regular function
        Name   *Name
        Type   *FuncType
        Body   *BlockStmt // nil means no body (forward declaration)
        Pragma Pragma     // TODO(mdempsky): Cleaner solution.
        decl
}
```

```
// SourceFile = PackageClause ";" { ImportDecl ";" } { TopLevelDecl ";" } .
func (p *parser) fileOrNil() *File {
        if trace {
                defer p.trace("file")()
        }

        f := new(File)
        f.pos = p.pos()

        // PackageClause
        if !p.got(_Package) {
                p.syntax_error("package statement must be first")
                return nil
        }
        f.PkgName = p.name()
        p.want(_Semi)

        // don't bother continuing if package clause has errors
        if p.first != nil {
                return nil
        }

        // { ImportDecl ";" }
        for p.got(_Import) {
                f.DeclList = p.appendGroup(f.DeclList, p.importDecl)
                p.want(_Semi)
        }

        // { TopLevelDecl ";" }
        for p.tok != _EOF {
                switch p.tok {
                case _Const:
                        p.next()
                        f.DeclList = p.appendGroup(f.DeclList, p.constDecl)

                case _Type:
                        p.next()
                        f.DeclList = p.appendGroup(f.DeclList, p.typeDecl)

                case _Var:
                        p.next()
                        f.DeclList = p.appendGroup(f.DeclList, p.varDecl)

                case _Func:
                        p.next()
                        if d := p.funcDeclOrNil(); d != nil {
                                f.DeclList = append(f.DeclList, d)
                        }

                default:
                        if p.tok == _Lbrace && len(f.DeclList) > 0 && isEmptyF
                                // opening { of function declaration on next l
                                p.syntax_error("unexpected semicolon or newlin
                        } else {
                                p.syntax_error("non-declaration statement outs
                        }
```

Statement

```go
type (
	Stmt interface {
		Node
		aStmt()
	}

	SimpleStmt interface {
		Stmt
		aSimpleStmt()
	}

	EmptyStmt struct {
		simpleStmt
	}

	LabeledStmt struct {
		Label *Name
		Stmt  Stmt
		stmt
	}

	BlockStmt struct {
		List   []Stmt
		Rbrace src.Pos
		stmt
	}

	ExprStmt struct {
		X Expr
		simpleStmt
	}

	SendStmt struct {
		Chan, Value Expr // Chan <- Value
		simpleStmt
	}

	DeclStmt struct {
		DeclList []Decl
		stmt
	}

	AssignStmt struct {
		Op       Operator // 0 means no operatio
		Lhs, Rhs Expr     // Rhs == ImplicitOne
		simpleStmt
	}
```

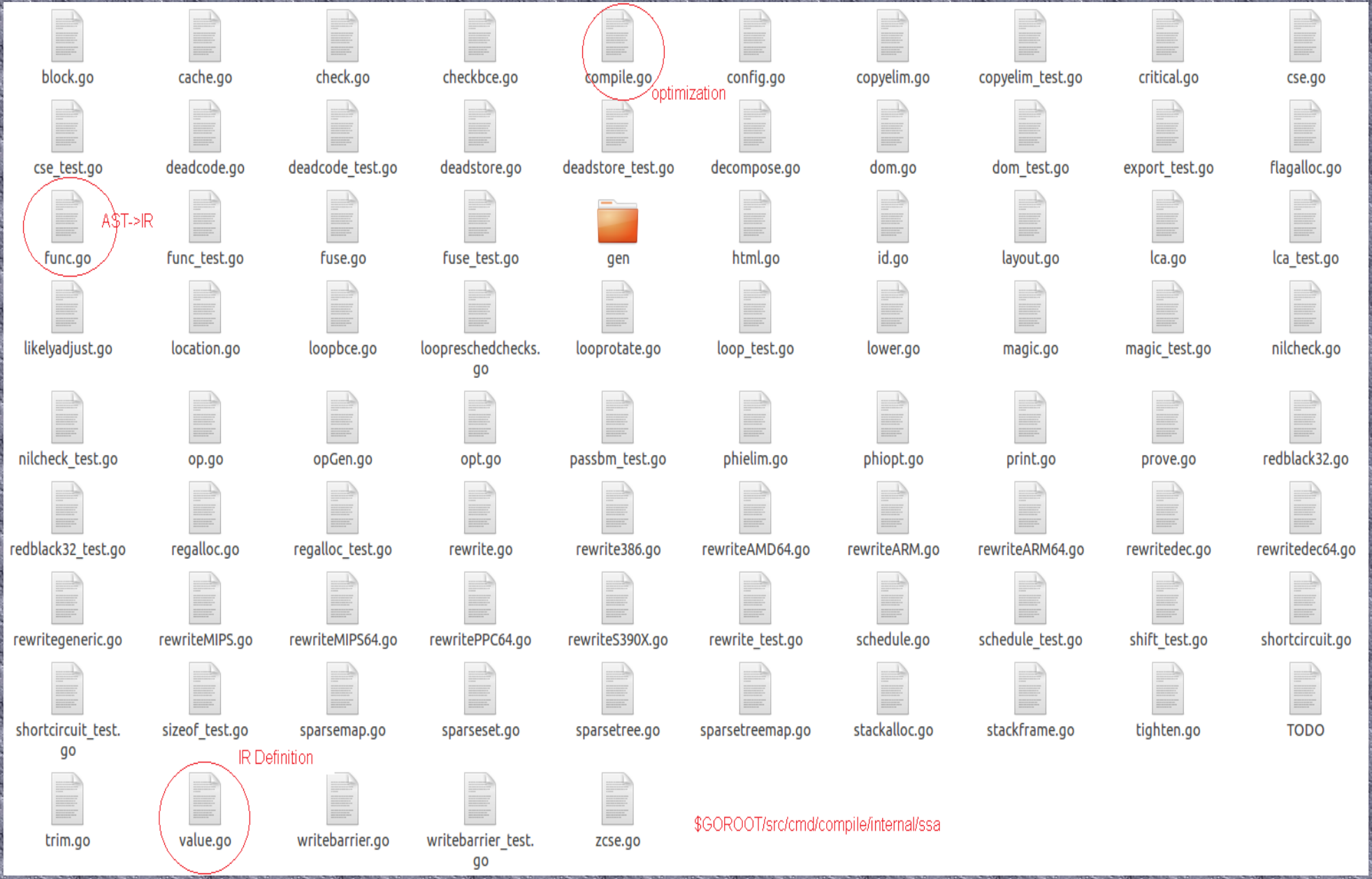
```go
// Statement =
//	Declaration | LabeledStmt | SimpleStmt |
//	GoStmt | ReturnStmt | BreakStmt | ContinueStmt | GotoStmt |
//	FallthroughStmt | Block | IfStmt | SwitchStmt | SelectStmt | ForStmt
//	DeferStmt .
func (p *parser) stmtOrNil() Stmt {
	if trace {
		defer p.trace("stmt " + p.tok.String())()
	}

	// Most statements (assignments) start with an identifier;
	// look for it first before doing anything more expensive.
	if p.tok == _Name {
		lhs := p.exprList()
		if label, ok := lhs.(*Name); ok && p.tok == _Colon {
			return p.labeledStmtOrNil(label)
		}
		return p.simpleStmt(lhs, false)
	}

	switch p.tok {
	case _Lbrace:
		return p.blockStmt("")

	case _Var:
		return p.declStmt(p.varDecl)

	case _Const:
		return p.declStmt(p.constDecl)

	case _Type:
		return p.declStmt(p.typeDecl)

	case _Operator, _Star:
		switch p.op {
		case Add, Sub, Mul, And, Xor, Not:
			return p.simpleStmt(nil, false) // unary operators
		}

	case _Literal, _Func, _Lparen, // operands
		_Lbrack, _Struct, _Map, _Chan, _Interface, // composite types
		_Arrow: // receive operator
		return p.simpleStmt(nil, false)

	case _For:
		return p.forStmt()

	case _Switch:
		return p.switchStmt()

	case _Select:
		return p.selectStmt()

	case _If:
		return p.ifStmt()
```

Expression

```go
// Key: Value
KeyValueExpr struct {
	Key, Value Expr
	expr
}

// func Type { Body }
FuncLit struct {
	Type *FuncType
	Body *BlockStmt
	expr
}

// (X)
ParenExpr struct {
	X Expr
	expr
}

// X.Sel
SelectorExpr struct {
	X   Expr
	Sel *Name
	expr
}

// X[Index]
IndexExpr struct {
	X     Expr
	Index Expr
	expr
}

// X[Index[0] : Index[1] : Index[2]]
SliceExpr struct {
	X     Expr
	Index [3]Expr
	// Full indicates whether this is a simple or full slic
	// In a valid AST, this is equivalent to Index[2] != ni
	// TODO(mdempsky): This is only needed to report the "3
	// slice of string" error when Index[2] is missing.
	Full bool
	expr
}

// X.(Type)
AssertExpr struct {
	X Expr
	// TODO(gri) consider using Name{"..."} instead of nil
	Type Expr
	expr
}
```

```go
func (p *parser) expr() Expr {
	if trace {
		defer p.trace("expr")()
	}

	return p.binaryExpr(0)
}

// Expression = UnaryExpr | Expression binary_op Expression .
func (p *parser) binaryExpr(prec int) Expr {
	// don't trace binaryExpr - only leads to overly nested trace output

	x := p.unaryExpr()
	for (p.tok == _Operator || p.tok == _Star) && p.prec > prec {
		t := new(Operation)
		t.pos = p.pos()
		t.Op = p.op
		t.X = x
		tprec := p.prec
		p.next()
		t.Y = p.binaryExpr(tprec)
		x = t
	}
	return x
}

// UnaryExpr = PrimaryExpr | unary_op UnaryExpr .
func (p *parser) unaryExpr() Expr {
	if trace {
		defer p.trace("unaryExpr")()
	}

	switch p.tok {
	case _Operator, _Star:
		switch p.op {
		case Mul, Add, Sub, Not, Xor:
			x := new(Operation)
			x.pos = p.pos()
			x.Op = p.op
			p.next()
			x.X = p.unaryExpr()
			return x

		case And:
			x := new(Operation)
			x.pos = p.pos()
			x.Op = And
			p.next()
			// unaryExpr may have returned a parenthesized composite literal
			// (see comment in operand) - remove parentheses if any
			x.X = unparen(p.unaryExpr())
			return x
		}
```

Middle End


block.go
cache.go
check.go
checkbce.go
compile.go
optimization
config.go
copyelim.go
copyelim_test.go
critical.go
cse.go
cse_test.go
deadcode.go
deadcode_test.go
deadstore.go
deadstore_test.go
decompose.go
dom.go
dom_test.go
export_test.go
flagalloc.go
AST->IR
func.go
func_test.go
fuse.go
fuse_test.go
gen
html.go
id.go
layout.go
lca.go
lca_test.go
likelyadjust.go
location.go
loopbce.go
loopreschedchecks.go
looprotate.go
loop_test.go
lower.go
magic.go
magic_test.go
nilcheck.go
nilcheck_test.go
op.go
opGen.go
opt.go
passbm_test.go
phielim.go
phiopt.go
print.go
prove.go
redblack32.go
redblack32_test.go
regalloc.go
regalloc_test.go
rewrite.go
rewrite386.go
rewriteAMD64.go
rewriteARM.go
rewriteARM64.go
rewritedec.go
rewritedec64.go
rewritegeneric.go
rewriteMIPS.go
rewriteMIPS64.go
rewritePPC64.go
rewriteS390X.go
rewrite_test.go
schedule.go
schedule_test.go
shift_test.go
shortcircuit.go
shortcircuit_test.go
sizeof_test.go
sparsemap.go
sparseset.go
sparsetree.go
sparsetreemap.go
stackalloc.go
stackframe.go
tighten.go
TODO
IR Definition
trim.go
value.go
writebarrier.go
writebarrier_test.go
zcse.go
$GOROOT/src/cmd/compile/internal/ssa

IR Definition

```go
// A Value represents a value in the SSA representation of the program.
// The ID and Type fields must not be modified. The remainder may be modified
// if they preserve the value of the Value (e.g. changing a (mul 2 x) to an (add x x)).
type Value struct {
	// A unique identifier for the value. For performance we allocate these IDs
	// densely starting at 1.  There is no guarantee that there won't be occasional holes, though.
	ID ID

	// The operation that computes this value. See op.go.
	Op Op

	// The type of this value. Normally this will be a Go type, but there
	// are a few other pseudo-types, see type.go.
	Type *types.Type

	// Auxiliary info for this value. The type of this information depends on the opcode and type.
	// AuxInt is used for integer values, Aux is used for other values.
	// Floats are stored in AuxInt using math.Float64bits(f).
	AuxInt int64
	Aux    interface{}

	// Arguments of this value
	Args []*Value

	// Containing basic block
	Block *Block

	// Source position
	Pos src.XPos

	// Use count. Each appearance in Value.Args and Block.Control counts once.
	Uses int32

	// Storage for the first three args
	argstorage [3]*Value
}
```

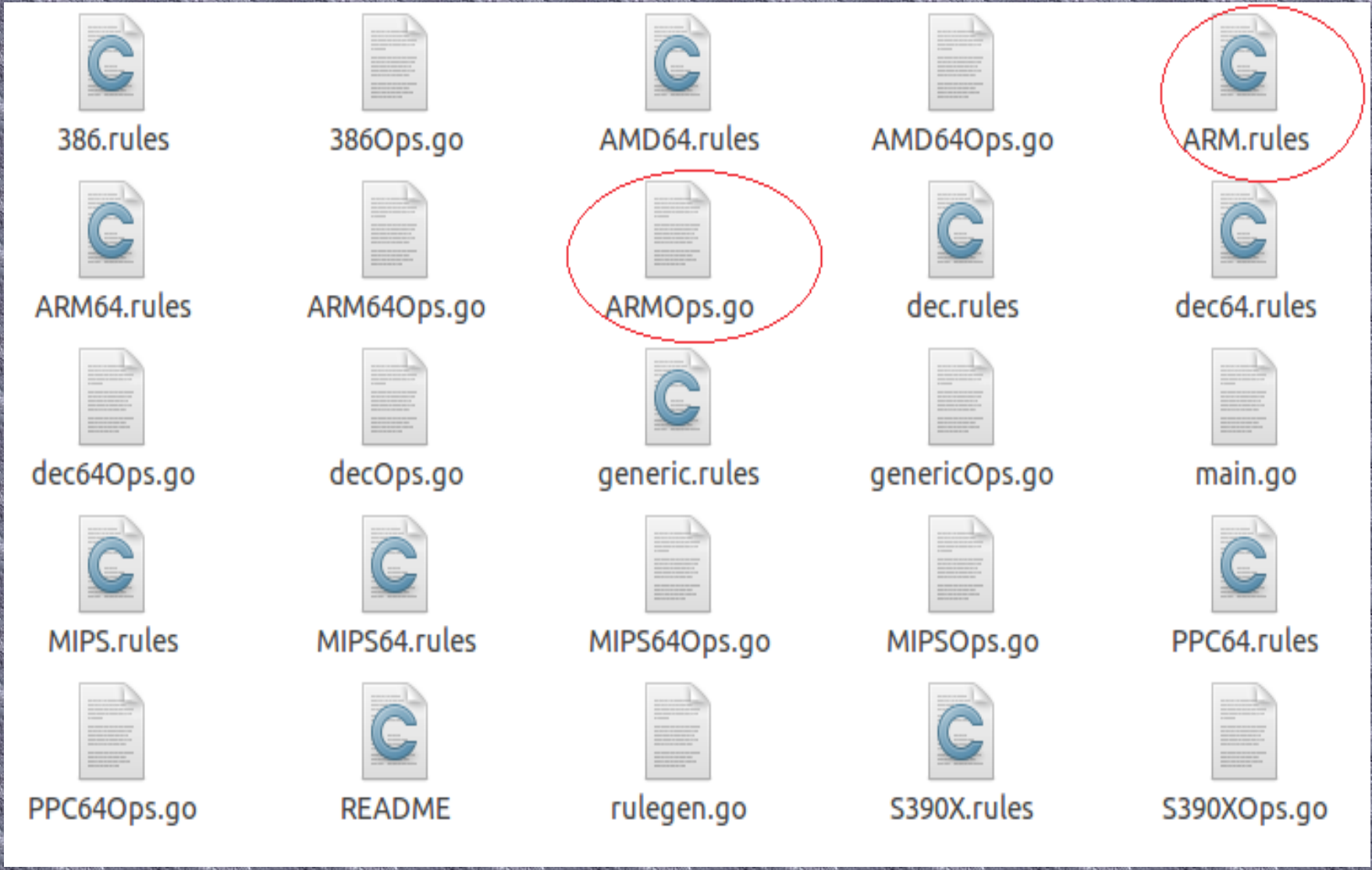
Generic Operator

```
// Unused portions of AuxInt are filled by sign-extending the used portion.
// Users of AuxInt which interpret AuxInt as unsigned (e.g. shifts) must be careful.
var genericOps = []opData{
        // 2-input arithmetic
        // Types must be consistent with Go typing. Add, for example, must take two values
        // of the same type and produces that same type.
        {name: "Add8", argLength: 2, commutative: true}, // arg0 + arg1
        {name: "Add16", argLength: 2, commutative: true},
        {name: "Add32", argLength: 2, commutative: true},
        {name: "Add64", argLength: 2, commutative: true},
        {name: "AddPtr", argLength: 2}, // For address calculations.  arg0 is a pointer and arg1 is an int.
        {name: "Add32F", argLength: 2, commutative: true},
        {name: "Add64F", argLength: 2, commutative: true},

        {name: "Sub8", argLength: 2}, // arg0 - arg1
        {name: "Sub16", argLength: 2},
        {name: "Sub32", argLength: 2},
        {name: "Sub64", argLength: 2},
        {name: "SubPtr", argLength: 2},
        {name: "Sub32F", argLength: 2},
        {name: "Sub64F", argLength: 2},

        {name: "Mul8", argLength: 2, commutative: true}, // arg0 * arg1
        {name: "Mul16", argLength: 2, commutative: true},
        {name: "Mul32", argLength: 2, commutative: true},
        {name: "Mul64", argLength: 2, commutative: true},
        {name: "Mul32F", argLength: 2, commutative: true},
        {name: "Mul64F", argLength: 2, commutative: true},

        {name: "Div32F", argLength: 2}, // arg0 / arg1
        {name: "Div64F", argLength: 2},

        {name: "Hmul32", argLength: 2, commutative: true},
        {name: "Hmul32u", argLength: 2, commutative: true},
        {name: "Hmul64", argLength: 2, commutative: true},
        {name: "Hmul64u", argLength: 2, commutative: true},

        {name: "Mul32uhilo", argLength: 2, typ: "(UInt32,UInt32)", commutative: true}, // arg0 * arg1, returns (hi, lo)
        {name: "Mul64uhilo", argLength: 2, typ: "(UInt64,UInt64)", commutative: true}, // arg0 * arg1, returns (hi, lo)
```

IR Transform

```
// list of passes for the compiler
var passes = [...]pass{
        // TODO: combine phielim and copyelim into a single pass?
        {name: "early phielim", fn: phielim},
        {name: "early copyelim", fn: copyelim},
        {name: "early deadcode", fn: deadcode}, // remove generated dead code to avoid doing pointless work during opt
        {name: "short circuit", fn: shortcircuit},
        {name: "decompose user", fn: decomposeUser, required: true},
        {name: "opt", fn: opt, required: true},                // TODO: split required rules and optimizing rules
        {name: "zero arg cse", fn: zcse, required: true},      // required to merge OpSB values
        {name: "opt deadcode", fn: deadcode, required: true}, // remove any blocks orphaned during opt
        {name: "generic cse", fn: cse},
        {name: "phiopt", fn: phiopt},
        {name: "nilcheckelim", fn: nilcheckelim},
        {name: "prove", fn: prove},
        {name: "loopbce", fn: loopbce},
        {name: "decompose builtin", fn: decomposeBuiltIn, required: true},
        {name: "dec", fn: dec, required: true},
        {name: "late opt", fn: opt, required: true}, // TODO: split required rules and optimizing rules
        {name: "generic deadcode", fn: deadcode},
        {name: "check bce", fn: checkbce},
        {name: "fuse", fn: fuse},
        {name: "dse", fn: dse},
        {name: "writebarrier", fn: writebarrier, required: true}, // expand write barrier ops
        {name: "insert resched checks", fn: insertLoopReschedChecks,
                disabled: objabi.Preemptibleloops_enabled == 0}, // insert resched checks in loops.
        {name: "tighten", fn: tighten}, // move values closer to their uses
        {name: "lower", fn: lower, required: true},
        {name: "lowered cse", fn: cse},
        {name: "lowered deadcode", fn: deadcode, required: true},
        {name: "checkLower", fn: checkLower, required: true},
        {name: "late phielim", fn: phielim},
        {name: "late copyelim", fn: copyelim},
        {name: "phi tighten", fn: phiTighten},
        {name: "late deadcode", fn: deadcode},
        {name: "critical", fn: critical, required: true}, // remove critical edges
        {name: "likelyadjust", fn: likelyadjust},
        {name: "layout", fn: layout, required: true},     // schedule blocks
        {name: "schedule", fn: schedule, required: true}, // schedule values
        {name: "late nilcheck", fn: nilcheckelim2},
        {name: "flagalloc", fn: flagalloc, required: true}, // allocate flags register
        {name: "regalloc", fn: regalloc, required: true},   // allocate int & float registers + stack slots
        {name: "loop rotate", fn: loopRotate},
        {name: "stackframe", fn: stackframe, required: true},
        {name: "trim", fn: trim}, // remove empty blocks
}
```

Back End


386.rules
386Ops.go
AMD64.rules
AMD64Ops.go
ARM.rules
ARM64.rules
ARM64Ops.go
ARMOps.go
dec.rules
dec64.rules
dec64Ops.go
decOps.go
generic.rules
genericOps.go
main.go
MIPS.rules
MIPS64.rules
MIPS64Ops.go
MIPSOps.go
PPC64.rules
PPC64Ops.go
README
rulegen.go
S390X.rules
S390XOps.go

Lower ARM Operator

```
ops := []opData{
        // binary ops
        {name: "ADD", argLength: 2, reg: gp21, asm: "ADD", commutative: true},     // arg0 + arg1
        {name: "ADDconst", argLength: 1, reg: gp11sp, asm: "ADD", aux: "Int32"},   // arg0 + auxInt
        {name: "SUB", argLength: 2, reg: gp21, asm: "SUB"},                        // arg0 - arg1
        {name: "SUBconst", argLength: 1, reg: gp11, asm: "SUB", aux: "Int32"},     // arg0 - auxInt
        {name: "RSB", argLength: 2, reg: gp21, asm: "RSB"},                        // arg1 - arg0
        {name: "RSBconst", argLength: 1, reg: gp11, asm: "RSB", aux: "Int32"},     // auxInt - arg0
        {name: "MUL", argLength: 2, reg: gp21, asm: "MUL", commutative: true},     // arg0 * arg1
        {name: "HMUL", argLength: 2, reg: gp21, asm: "MULL", commutative: true},   // (arg0 * arg1) >> 32, signed
        {name: "HMULU", argLength: 2, reg: gp21, asm: "MULLU", commutative: true}, // (arg0 * arg1) >> 32, unsigned

        // udiv runtime call for soft division
        // output0 = arg0/arg1, output1 = arg0%arg1
        // see ../../../../../runtime/vlop_arm.s
        {
                name:      "CALLudiv",
                argLength: 2,
                reg: regInfo{
                        inputs:   []regMask{buildReg("R1"), buildReg("R0")},
                        outputs:  []regMask{buildReg("R0"), buildReg("R1")},
                        clobbers: buildReg("R2 R3 R14"), // also clobbers R12 on NaCl (modified in ../config.go)
                },
                clobberFlags: true,
                typ:          "(UInt32,UInt32)",
                call:         false, // TODO(mdempsky): Should this be true?
        },

        {name: "ADDS", argLength: 2, reg: gp21carry, asm: "ADD", commutative: true}, // arg0 + arg1, set carry flag
        {name: "ADDSconst", argLength: 1, reg: gp11carry, asm: "ADD", aux: "Int32"}, // arg0 + auxInt, set carry flag
        {name: "ADC", argLength: 3, reg: gp2flags1, asm: "ADC", commutative: true},  // arg0 + arg1 + carry, arg2=flags
        {name: "ADCconst", argLength: 2, reg: gp1flags1, asm: "ADC", aux: "Int32"},  // arg0 + auxInt + carry, arg1=flags
        {name: "SUBS", argLength: 2, reg: gp21carry, asm: "SUB"},                    // arg0 - arg1, set carry flag
        {name: "SUBSconst", argLength: 1, reg: gp11carry, asm: "SUB", aux: "Int32"}, // arg0 - auxInt, set carry flag
        {name: "RSBSconst", argLength: 1, reg: gp11carry, asm: "RSB", aux: "Int32"}, // auxInt - arg0, set carry flag
        {name: "SBC", argLength: 3, reg: gp2flags1, asm: "SBC"},                      // arg0 - arg1 - carry, arg2=flags
        {name: "SBCconst", argLength: 2, reg: gp1flags1, asm: "SBC", aux: "Int32"},  // arg0 - auxInt - carry, arg1=flags
        {name: "RSCconst", argLength: 2, reg: gp1flags1, asm: "RSC", aux: "Int32"},  // auxInt - arg0 - carry, arg1=flags

        {name: "MULLU", argLength: 2, reg: gp22, asm: "MULLU", commutative: true}, // arg0 * arg1, high 32 bits in out0, low 32 bits in out1
        {name: "MULA", argLength: 3, reg: gp31, asm: "MULA"},                      // arg0 * arg1 + arg2
```

IR -> ARM Assembly

```
(AddPtr x y) -> (ADD x y)
(Add32 x y) -> (ADD x y)
(Add16 x y) -> (ADD x y)
(Add8 x y) -> (ADD x y)
(Add32F x y) -> (ADDF x y)
(Add64F x y) -> (ADDD x y)

(Add32carry x y) -> (ADDS x y)
(Add32withcarry x y c) -> (ADC x y c)

(SubPtr x y) -> (SUB x y)
(Sub32 x y) -> (SUB x y)
(Sub16 x y) -> (SUB x y)
(Sub8 x y) -> (SUB x y)
(Sub32F x y) -> (SUBF x y)
(Sub64F x y) -> (SUBD x y)

(Sub32carry x y) -> (SUBS x y)
(Sub32withcarry x y c) -> (SBC x y c)

(Mul32 x y) -> (MUL x y)
(Mul16 x y) -> (MUL x y)
(Mul8 x y) -> (MUL x y)
(Mul32F x y) -> (MULF x y)
(Mul64F x y) -> (MULD x y)

(Hmul32 x y) -> (HMUL x y)
(Hmul32u x y) -> (HMULU x y)

(Mul32uhilo x y) -> (MULLU x y)

(Div32 x y) ->
        (SUB (XOR <typ.UInt32>                                                    // negate the result if one op
                (Select0 <typ.UInt32> (CALLudiv
                        (SUB <typ.UInt32> (XOR x <typ.UInt32> (Signmask x)) (Signmask x))   // negate x if negative
                        (SUB <typ.UInt32> (XOR y <typ.UInt32> (Signmask y)) (Signmask y)))) // negate y if negative
                (Signmask (XOR <typ.UInt32> x y))) (Signmask (XOR <typ.UInt32> x y)))
(Div32u x y) -> (Select0 <typ.UInt32> (CALLudiv x y))
(Div16 x y) -> (Div32 (SignExt16to32 x) (SignExt16to32 y))
(Div16u x y) -> (Div32u (ZeroExt16to32 x) (ZeroExt16to32 y))
(Div8 x y) -> (Div32 (SignExt8to32 x) (SignExt8to32 y))
(Div8u x y) -> (Div32u (ZeroExt8to32 x) (ZeroExt8to32 y))
```

Peep Hole Optimization

```
// generic constant folding
(ADDconst [c] x) && !isARMImmRot(uint32(c)) && isARMImmRot(uint32(-c)) -> (SUBconst [int64(int32(-c))] x)
(SUBconst [c] x) && !isARMImmRot(uint32(c)) && isARMImmRot(uint32(-c)) -> (ADDconst [int64(int32(-c))] x)
(ANDconst [c] x) && !isARMImmRot(uint32(c)) && isARMImmRot(^uint32(c)) -> (BICconst [int64(^uint32(c))] x)
(BICconst [c] x) && !isARMImmRot(uint32(c)) && isARMImmRot(^uint32(c)) -> (ANDconst [int64(^uint32(c))] x)
(ADDconst [c] (MOVWconst [d])) -> (MOVWconst [int64(int32(c+d))])
(ADDconst [c] (ADDconst [d] x)) -> (ADDconst [int64(int32(c+d))] x)
(ADDconst [c] (SUBconst [d] x)) -> (ADDconst [int64(int32(c-d))] x)
(ADDconst [c] (RSBconst [d] x)) -> (RSBconst [int64(int32(c+d))] x)
(ADCconst [c] (ADDconst [d] x) flags) -> (ADCconst [int64(int32(c+d))] x flags)
(ADCconst [c] (SUBconst [d] x) flags) -> (ADCconst [int64(int32(c-d))] x flags)
(SUBconst [c] (MOVWconst [d])) -> (MOVWconst [int64(int32(d-c))])
(SUBconst [c] (SUBconst [d] x)) -> (ADDconst [int64(int32(-c-d))] x)
(SUBconst [c] (ADDconst [d] x)) -> (ADDconst [int64(int32(-c+d))] x)
(SUBconst [c] (RSBconst [d] x)) -> (RSBconst [int64(int32(-c+d))] x)
(SBCconst [c] (ADDconst [d] x) flags) -> (SBCconst [int64(int32(c-d))] x flags)
(SBCconst [c] (SUBconst [d] x) flags) -> (SBCconst [int64(int32(c+d))] x flags)
(RSBconst [c] (MOVWconst [d])) -> (MOVWconst [int64(int32(c-d))])
(RSBconst [c] (RSBconst [d] x)) -> (ADDconst [int64(int32(c-d))] x)
(RSBconst [c] (ADDconst [d] x)) -> (RSBconst [int64(int32(c-d))] x)
(RSBconst [c] (SUBconst [d] x)) -> (RSBconst [int64(int32(c+d))] x)
(RSCconst [c] (ADDconst [d] x) flags) -> (RSCconst [int64(int32(c-d))] x flags)
(RSCconst [c] (SUBconst [d] x) flags) -> (RSCconst [int64(int32(c+d))] x flags)
(SLLconst [c] (MOVWconst [d])) -> (MOVWconst [int64(uint32(d)<<uint64(c))])
(SRLconst [c] (MOVWconst [d])) -> (MOVWconst [int64(uint32(d)>>uint64(c))])
(SRAconst [c] (MOVWconst [d])) -> (MOVWconst [int64(int32(d)>>uint64(c))])
(MUL (MOVWconst [c]) (MOVWconst [d])) -> (MOVWconst [int64(int32(c*d))])
(MULA (MOVWconst [c]) (MOVWconst [d]) a) -> (ADDconst [int64(int32(c*d))] a)
(Select0 (CALLudiv (MOVWconst [c]) (MOVWconst [d]))) -> (MOVWconst [int64(uint32(c)/uint32(d))])
(Select1 (CALLudiv (MOVWconst [c]) (MOVWconst [d]))) -> (MOVWconst [int64(uint32(c)%uint32(d))])
(ANDconst [c] (MOVWconst [d])) -> (MOVWconst [c&d])
(ANDconst [c] (ANDconst [d] x)) -> (ANDconst [c&d] x)
(ORconst [c] (MOVWconst [d])) -> (MOVWconst [c|d])
(ORconst [c] (ORconst [d] x)) -> (ORconst [c|d] x)
(XORconst [c] (MOVWconst [d])) -> (MOVWconst [c^d])
(XORconst [c] (XORconst [d] x)) -> (XORconst [c^d] x)
(BICconst [c] (MOVWconst [d])) -> (MOVWconst [d&^c])
(BICconst [c] (BICconst [d] x)) -> (BICconst [int64(int32(c|d))] x)
```

Assembler

```
var optab = []Optab{
        /* struct Optab:
        OPCODE, from, prog->reg, to,                type,size,param,flag */
        {obj.ATEXT, C_ADDR, C_NONE, C_TEXTSIZE, 0, 0, 0, 0, 0},
        {AADD, C_REG, C_REG, C_REG, 1, 4, 0, 0, 0},
        {AADD, C_REG, C_NONE, C_REG, 1, 4, 0, 0, 0},
        {AAND, C_REG, C_REG, C_REG, 1, 4, 0, 0, 0},
        {AAND, C_REG, C_NONE, C_REG, 1, 4, 0, 0, 0},
        {AORR, C_REG, C_REG, C_REG, 1, 4, 0, 0, 0},
        {AORR, C_REG, C_NONE, C_REG, 1, 4, 0, 0, 0},
        {AMOVW, C_REG, C_NONE, C_REG, 1, 4, 0, 0, 0},
        {AMVN, C_REG, C_NONE, C_REG, 1, 4, 0, 0, 0},
        {ACMP, C_REG, C_REG, C_NONE, 1, 4, 0, 0, 0},
        {AADD, C_RCON, C_REG, C_REG, 2, 4, 0, 0, 0},
        {AADD, C_RCON, C_NONE, C_REG, 2, 4, 0, 0, 0},
        {AAND, C_RCON, C_REG, C_REG, 2, 4, 0, 0, 0},
        {AAND, C_RCON, C_NONE, C_REG, 2, 4, 0, 0, 0},
        {AORR, C_RCON, C_REG, C_REG, 2, 4, 0, 0, 0},
        {AORR, C_RCON, C_NONE, C_REG, 2, 4, 0, 0, 0},
        {AMOVW, C_RCON, C_NONE, C_REG, 2, 4, 0, 0, 0},
        {AMVN, C_RCON, C_NONE, C_REG, 2, 4, 0, 0, 0},
        {ACMP, C_RCON, C_REG, C_NONE, 2, 4, 0, 0, 0},
        {AADD, C_SHIFT, C_REG, C_REG, 3, 4, 0, 0, 0},
        {AADD, C_SHIFT, C_NONE, C_REG, 3, 4, 0, 0, 0},
        {AAND, C_SHIFT, C_REG, C_REG, 3, 4, 0, 0, 0},
        {AAND, C_SHIFT, C_NONE, C_REG, 3, 4, 0, 0, 0},
```

My Work on Golang

Total 29 commits / 14,232 lines. Mainly ARM compiler, assembler and library.

owner:powerman1st@163.com · Gerrit Code Review - Mozilla Firefox

https://go-review.googlesource.com/q/owner:powerman1st%2540163.com

PolyGerrit Changes Documentation Browse owner:powerman1st@163.com Search Sign in

Subject	Status	Owner	Project	Branch	Updated	Size	CR	RT	TR
cmd/internal/obj/arm: support more ARMv6 instructions		Ben Shi	go	master	Sep 28	+148, -6			
cmd/compile: optimized ARM code with BFX/BFXU	Merged	Ben Shi	go	master	Sep 21	+125, -0	✓	✓	✗
cmd/compile: optimize ARM code with MULAF/MULSF/MULAD/MULSD	Merged	Ben Shi	go	master	Sep 16	+382, -0	+1		
cmd/compile: optimize ARM code with NMULF/NMULD	Merged	Ben Shi	go	master	Sep 11	+285, -9	+1		
cmd/internal/obj/arm: support more ARM VFP instructions	Merged	Ben Shi	go	master	Sep 11	+58, -1	✓	✓	✓
cmd/internal/obj/arm: support more ARM VFP instructions	Merged	Ben Shi	go	master	Aug 31	+76, -1	✓	✓	✓
cmd/compile: optimize ARM with MULS	Merged	Ben Shi	go	master	Aug 30	+570, -1	✓	✓	✗
cmd/compile: optimize ARM with more efficient MOVB/MOVBU/MOVH/MOVHU	Merged	Ben Shi	go	master	Aug 29	+650, -2	+1		
cmd/internal/obj/arm: support BFX/BFXU instructions	Merged	Ben Shi	go	master	Aug 22	+85, -12	+1		
cmd/internal/obj/arm: support new arm instructions	Merged	Ben Shi	go	master	Aug 18	+68, -4	+1		
arm/armasm/testdata: add more decoding tests	Merged	Ben Shi	arch	master	Jul 25	+324, -26	✓	✓	
arm: support MSR instruction in the disassembler	Merged	Ben Shi	arch	master	Jul 15	+42, -0	✓	✓	✓
arm/armasm: rename VLDR/VSTR in plan9 syntax	Merged	Ben Shi	arch	master	Jul 15	+72, -4	+1		
arm: support SDIV/UDIV in the disassembler	Merged	Ben Shi	arch	master	Jul 11	+74, -0	✓	✓	✓
arm/armasm: fix wrong register order in SMLABT/SMLATB/SMLATT/SMLAD/SMLSD	Merged	Ben Shi	arch	master	Jul 10	+8, -1	✓	✓	✓
cmd/internal/obj/arm: check illegal base registers in ARM instructions	Merged	Ben Shi	go	master	Jul 01	+47, -0	+1		
arm/armasm: fix wrong decoding MOVW/MOVH/MOVB to plan9 syntax	Merged	Ben Shi	arch	master	Jul 01	+921, -8	+1		
cmd/internal/obj/arm: fix wrong encoding of MULBB	Merged	Ben Shi	go	master	Jun 24	+2, -2	✓	✓	✓
cmd/internal/obj/arm: fix setting U bit in shifted register offset of MOVBS	Merged	Ben Shi	go	master	Jun 23	+398, -7	+1		
arm/armasm: fix wrong register order in MLA.S	Merged	Ben Shi	arch	master	Jun 22	+2, -1	✓	✓	✓
cmd/internal/obj/arm: fix MOVW to/from FPSR	Merged	Ben Shi	go	master	Jun 13	+98, -19	✓	✓	✓
cmd/internal/obj/arm: fix encoding of move register/immediate to CPSR	Merged	Ben Shi	go	master	Jun 09	+32, -3	✓		
cmd/asm: fix operand order of ARM's MULA instruction	Merged	Ben Shi	go	master	Jun 06	+11, -11	✓	✓	✓
cmd/internal/obj/arm: fix constant decomposition	Merged	Ben Shi	go	master	Jun 06	+127, -1	+1		
cmd/asm/internal/asm: fix a bug in ARM assembly encoding test	Merged	Ben Shi	go	master	May 26	+56, -10	+1		

Next →

Powered by Gerrit Code Review (2.14.2-3498-g78d2cce4f7)

Send feedback | Switch to Old UI | Press "?" for keyboard shortcuts

ARM’s Hardware Divider

A8.8.248 UDIV

Unsigned Divide divides a 32-bit unsigned integer register value by a 32-bit unsigned integer register value, and writes the result to the destination register. The condition flags are not affected.

See *ARMv7 implementation requirements and options for the divide instructions* on page A4-172 for more information about this instruction.

Encoding T1 ARMv7-R, ARMv7VE, otherwise OPTIONAL in ARMv7-A

```
UDIV<c> <Rd>, <Rn>, <Rm>
```

15 14 13 12 11	10 9 8 7 6 5	4	3 2 1 0	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
1 1 1 1 1	0 1 1 1 0 1	1	Rn	(1) (1) (1) (1)	Rd	1 1 1 1	Rm

```
d = UInt(Rd);  n = UInt(Rn);  m = UInt(Rm);
if d IN {13,15} || n IN {13,15} || m IN {13,15} then UNPREDICTABLE;
```

Encoding A1 ARMv7VE, otherwise OPTIONAL in ARMv7-A and ARMv7-R

```
UDIV<c> <Rd>, <Rn>, <Rm>
```

31 30 29 28	27 26 25 24	23	22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7	6 5 4	3 2 1 0
cond	0 1 1 1	0	0 1 1	Rd	(1) (1) (1) (1)	Rm	0	0 0	1	Rn

For the case when `cond` is `0b1111`, see *Unconditional instructions* on page A5-216.

```
d = UInt(Rd);  n = UInt(Rn);  m = UInt(Rm);
if d == 15 || n == 15 || m == 15 then UNPREDICTABLE;
```

GCC Does It in Build Time

```
unsigned int div(unsigned int a, unsigned int b)
{
        return a / b;
}
```

```
apuser@tj07598pcu:~/Desktop$ arm-linux-gnueabihf-gcc b.c -O2 -Wall -S  -marm -march=armv7-a
apuser@tj07598pcu:~/Desktop$ cat b.s
        .arch armv7-a
        .eabi_attribute 28, 1
        .fpu vfpv3-d16
        .eabi_attribute 20, 1
        .eabi_attribute 21, 1
        .eabi_attribute 23, 3
        .eabi_attribute 24, 1
        .eabi_attribute 25, 1
        .eabi_attribute 26, 2
        .eabi_attribute 30, 2
        .eabi_attribute 34, 1
        .eabi_attribute 18, 4
        .file   "b.c"
        .global __aeabi_uidiv
        .text
        .align  2
        .global div
        .syntax unified
        .arm
        .type   div, %function
div:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 0, uses_anonymous_args = 0
        push    {r4, lr}
        bl      __aeabi_uidiv
        pop     {r4, pc}
        .size   div, .-div
        .ident  "GCC: (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
        .section        .note.GNU-stack,"",%progbits
```

```
apuser@tj07598pcu:~/Desktop$ arm-linux-gnueabihf-gcc b.c -O2 -Wall -S  -marm -march=armv7ve
apuser@tj07598pcu:~/Desktop$
apuser@tj07598pcu:~/Desktop$ cat b.s
        .arch armv7-a
        .arch_extension virt
        .arch_extension idiv
        .arch_extension sec
        .arch_extension mp
        .eabi_attribute 28, 1
        .fpu vfpv3-d16
        .eabi_attribute 20, 1
        .eabi_attribute 21, 1
        .eabi_attribute 23, 3
        .eabi_attribute 24, 1
        .eabi_attribute 25, 1
        .eabi_attribute 26, 2
        .eabi_attribute 30, 2
        .eabi_attribute 34, 1
        .eabi_attribute 18, 4
        .file   "b.c"
        .text
        .align  2
        .global div
        .syntax unified
        .arm
        .type   div, %function
div:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 0, uses_anonymous_args = 0
        @ link register save eliminated.
        udiv    r0, r0, r1
        bx      lr
        .size   div, .-div
        .ident  "GCC: (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
        .section        .note.GNU-stack,"",%progbits
        .arm
        .type   div, %function
div:
        @ args = 0, pretend = 0, frame = 0
        @ frame_needed = 0, uses_anonymous_args = 0
        @ link register save eliminated.
        udiv    r0, r0, r1
        bx      lr
        .size   div, .-div
        .ident  "GCC: (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609"
        .section        .note.GNU-stack,"",%progbits
```

Golang Does It in Run Time

```
NAME
       getauxval - retrieve a value from the auxiliary vector

SYNOPSIS
       #include <sys/auxv.h>

       unsigned long getauxval(unsigned long type);

DESCRIPTION
       The  getauxval() function retrieves values from the auxiliary vector, a
       mechanism that the kernel's ELF binary  loader  uses  to  pass  certain
       information to user space when a program is executed.
```

```
AT_HWCAP
       A  pointer  to  a multibyte mask of bits whose settings indicate
       detailed processor capabilities.  The contents of the  bit  mask
       are  hardware dependent (for example, see the kernel source file
       arch/x86/include/asm/cpufeature.h for details  relating  to  the
       Intel  x86  architecture).  A human-readable version of the same
       information is available via /proc/cpuinfo.
```

```
processor       : 3
model name      : ARMv7 Processor rev 5 (v7l)
BogoMIPS        : 38.40
Features        : half thumb fastmult vfp edsp neon vfpv3 tls vfpv4 idiva idivt vfpd32 lpae evtstrm
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xc07
CPU revision    : 5
```

```
/* The following must match the kernel's
#define HWCAP_ARM_SWP           1
#define HWCAP_ARM_HALF          2
#define HWCAP_ARM_THUMB         4
#define HWCAP_ARM_26BIT         8
#define HWCAP_ARM_FAST_MULT     16
#define HWCAP_ARM_FPA           32
#define HWCAP_ARM_VFP           64
#define HWCAP_ARM_EDSP          128
#define HWCAP_ARM_JAVA          256
#define HWCAP_ARM_IWMMXT        512
#define HWCAP_ARM_CRUNCH        1024
#define HWCAP_ARM_THUMBEE       2048
#define HWCAP_ARM_NEON          4096
#define HWCAP_ARM_VFPv3         8192
#define HWCAP_ARM_VFPv3D16      16384
#define HWCAP_ARM_TLS           32768
#define HWCAP_ARM_VFPv4         65536
#define HWCAP_ARM_IDIVA         131072
#define HWCAP_ARM_IDIVT         262144
#define HWCAP_ARM_TLS           32768
#define HWCAP_ARM_VFPv4         65536
#define HWCAP_ARM_IDIVA         131072
#define HWCAP_ARM_IDIVT         262144
```

My Optimization Commit

golang / go

Watch 2,336 | Star 30,538 | Fork 4,120

Code | Issues 2,983 | Pull requests 0 | Wiki | Insights

runtime: use hardware divider to improve performance

Browse files

```
The hardware divider is an optional component of ARMv7. This patch
detects whether it is available in runtime and use it or not.

1. The hardware divider is detected at startup and a flag is set/clear
   according to a perticular bit of runtime.hwcap.
2. Each call of runtime.udiv will check this flag and decide if
   use the hardware division instruction.

A rough test shows the performance improves 40-50% for ARMv7. And
the compatibility of ARMv5/v6 is not broken.

fixes #19118

Change-Id: Ic586bc9659ebc169553ca2004d2bdb721df823ac
Reviewed-on: https://go-review.googlesource.com/37496
Run-TryBot: Cherry Zhang <cherryyz@google.com>
TryBot-Result: Gobot Gobot <gobot@golang.org>
Reviewed-by: Cherry Zhang <cherryyz@google.com>
```

master (#1) go1.9rc1 ··· go1.9beta1

benshi001 committed with **cherrymui** on 27 Feb 1 parent 2a8d99e commit 69261ecad6dd2f3efd5e4a249325ea27311526b6

My Optimization Commit

3 src/runtime/os_linux_arm.go

```
@@ -11,11 +11,13 @@ const (

        _HWCAP_VFP   = 1 << 6  // introduced in at least 2.6.11
        _HWCAP_VFPv3 = 1 << 13 // introduced in 2.6.30
+       _HWCAP_IDIVA = 1 << 17
 )

 var randomNumber uint32
 var armArch uint8 = 6 // we default to ARMv6
 var hwcap uint32      // set by setup_auxv
+var hardDiv bool      // set if a hardware divider is available

 func checkgoarm() {
        // On Android, /proc/self/auxv might be unreadable and hwcap won't
@@ -53,6 +55,7 @@ func archauxv(tag, val uintptr) {

        case _AT_HWCAP: // CPU capability bit flags
                hwcap = uint32(val)
+               hardDiv = (hwcap & _HWCAP_IDIVA) != 0
        }
 }
```

12 src/runtime/vlop_arm.s

```
@@ -119,6 +119,10 @@ TEXT runtime·_sfloatpanic(SB),NOSPLIT,$-4

 // Be careful: Ra == R11 will be used by the linker for synthesized instructions.
 TEXT udiv(SB),NOSPLIT,$-4
+       MOVBU   runtime·hardDiv(SB), Ra
+       CMP     $0, Ra
+       BNE     udiv_hardware
+
        CLZ     Rq, Rs // find normalizing shift
        MOVW.S  Rq<<Rs, Ra
        MOVW    $fast_udiv_tab<>-64(SB), RM
@@ -154,6 +158,14 @@ TEXT udiv(SB),NOSPLIT,$-4
        ADD.PL  $2, Rq
        RET

+// use hardware divider
+udiv_hardware:
+       DIVUHW  Rq, Rr, Rs
+       MUL     Rs, Rq, RM
+       RSB     Rr, RM, Rr
+       MOVW    Rs, Rq
+       RET
+
 udiv_by_large_d:
```

Constant De-folding

The encoding of a modified immediate constant in an ARM instruction is:

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12	11 10 9 8	7	6	5	4	3	2	1	0
	rotation	a	b	c	d	e	f	g	h

Table A5-6 shows the range of modified immediate constants available in ARM data-processing instructions, and their encoding in the a, b, c, d, e, f, g, and h bits and the rotation field in the instruction.

Table A5-6 Encoding of modified immediates in ARM processing instructions

rotation	<const> [a]
0000	00000000 00000000 00000000 abcdefgh
0001	gh000000 00000000 00000000 00abcdef
0010	efgh0000 00000000 00000000 0000abcd
0011	cdefgh00 00000000 00000000 000000ab
0100	abcdefgh 00000000 00000000 00000000
. . .	. . 8-bit values shifted to other even-numbered positions .
1001	00000000 00abcdef gh000000 00000000
. . .	. . 8-bit values shifted to other even-numbered positions .
1110	00000000 00000000 0000abcd efgh0000
1111	00000000 00000000 000000ab cdefgh00

GCC 5.4 vs Golang 1.9

a + 0x024ff230 -> a + 0x024c0000 + 0x0003f000 + 0x00000230

```
apuser@tj07598pcu: ~/Desktop
apuser@tj07598pcu: ~/Desktop        apuser@tj07598pcu: ~/Desktop
        @ link register save eliminated.
        add     r0, r0, #38535168
        add     r0, r0, #258048
        add     r0, r0, #560
        bx      lr
        .size   aa, .-aa
a.s                                                     23,2-9        91%
int aa(int a)
{
        return a + 0x024ff230;
}
~
a.c                                                     1,1           All
```

a + 0x024ff230 -> a + 0x02500000 – 0x00000dd0

```
apuser@tj07598pcu: ~/Desktop
apuser@tj07598pcu: ~/Desktop        apuser@tj07598pcu: ~/Desktop
  a.go:14       0x9d62c         e59d0028        MOVW 0x28(R13), R0
  a.go:15       0x9d630         e5900000        MOVW (R0), R0
  a.go:9        0x9d634         e2800625        ADD $38797312, R0, R0
  a.go:9        0x9d638         e2400edd        SUB $3536, R0, R0
  a.go:15       0x9d63c         e58d0024        MOVW R0, 0x24(R13)
a.txt [+]                                       147428,3      99%

func add(a int) int {
        return a + 0x024ff230;
}

a.go [+]                                        7,1           37%
-- INSERT --
```

My Optimization Commit

golang / go

Watch 2,339 | Star 30,565 | Fork 4,120

Code | Issues 2,988 | Pull requests 0 | Wiki | Insights

cmd/internal/obj: continue to optimize ARM's constant pool

Browse files

```
Both Keith's https://go-review.googlesource.com/c/41612/ and
and Ben's https://go-review.googlesource.com/c/41679/ optimized ARM's
constant pool. But neither was complete.

First, BIC was forgotten.
1. "BIC $0xff00ff00, Reg" can be optimized to
   "BIC $0xff000000, Reg
    BIC $0x0000ff00, Reg"
2. "BIC $0xffff00ff, Reg" can be optimized to
   "AND $0x0000ff00, Reg"
3. "AND $0xffff00ff, Reg" can be optimized to
   "BIC $0x0000ff00, Reg"

Second, break a non-ARMImmRot to the subtraction of two ARMImmRots was
left as TODO.
1. "ADD $0x00fffff0, Reg" can be optimized to
   "ADD $0x01000000, Reg
    SUB $0x00000010, Reg"
2. "SUB $0x00fffff0, Reg" can be optimized to
   "SUB $0x01000000, Reg
    ADD $0x00000010, Reg"

This patch fixes them and issue #19844.

The go1 benchmark shows improvements.

Change-Id: I5ad16cc0b29267bb4579aca3dcc10a0b8ade1aa4
Reviewed-on: https://go-review.googlesource.com/42430
Run-TryBot: Cherry Zhang <cherryyz@google.com>
TryBot-Result: Gobot Gobot <gobot@golang.org>
Reviewed-by: Cherry Zhang <cherryyz@google.com>
```

master (#17) go1.9rc1 ··· go1.9beta1

benshi001 committed with **cherrymui** on 2 May 1 parent 19b05ac commit 6897030fe3de43bbed48adb72f21a6c2d00042cd

My Optimization Commit

```
+// immrot2s returns bits encoding the immediate constant fields of two instructions,
+// such that the encoded constants y, x satisfy y-x==v, y&x==0.
+// Returns 0,0 if no such decomposition of v exists.
+func immrot2s(v uint32) (uint32, uint32) {
+       if immrot(v) == 0 {
+               return v, 0
+       }
+       // suppose v in the form of {leading 00, upper effective bits, lower 8 effective bits, trailing 00}
+       // omit trailing 00
+       var i uint32
+       for i = 2; i < 32; i += 2 {
+               if v&(1<<i-1) != 0 {
+                       break
+               }
+       }
+       // i must be <= 24, then adjust i just above lower 8 effective bits of v
+       i += 6
+       // let x = {the complement of lower 8 effective bits, trailing 00}, y = x + v
+       x := 1<<i - v&(1<<i-1)
+       y := v + x
+       if y, x = uint32(immrot(y)), uint32(immrot(x)); y != 0 && x != 0 {
+               return y, x
+       }
+       return 0, 0
+}
+
 func immaddr(v int32) int32 {
        if v >= 0 && v <= 0xfff {
                return v&0xfff | 1<<24 | 1<<23 /* pre indexing */ /* pre indexing, up */
@@ -1159,8 +1198,11 @@ func (c *ctxt5) aclass(a *obj.Addr) int {
```

0000	0010	0100	1111	1111	0010	0011	0000	024FF230
0000	0000	0000	0000	0000	0010	0011	0000	00000230
0000	0000	0000	0000	0000	1101	1101	0000	00000dd0
0000	0010	0101	0000	0000	0000	0000	0000	02500000

My Optimization Commit

```
+        case 107: /* op $I,R,R where I can be decomposed into 2 immediates */
+                c.aclass(&p.From)
+                r := int(p.Reg)
+                rt := int(p.To.Reg)
+                y, x := immrot2s(uint32(c.instoffset))
+                var as2 obj.As
+                switch p.As {
+                case AADD:
+                        as2 = ASUB // ADD -> ADD/SUB pair
+                case ASUB:
+                        as2 = AADD // SUB -> SUB/ADD pair
+                case ARSB:
+                        as2 = ASUB // RSB -> RSB/SUB pair
+                case AADC:
+                        as2 = ASUB // ADC -> ADC/SUB pair
+                case ASBC:
+                        as2 = AADD // SBC -> SBC/ADD pair
+                case ARSC:
+                        as2 = ASUB // RSC -> RSC/SUB pair
+                default:
+                        c.ctxt.Diag("unknown second op for %v", p)
+                }
+                o1 = c.oprrr(p, p.As, int(p.Scond))
+                o2 = c.oprrr(p, as2, int(p.Scond))
+                o1 |= (uint32(r)&15)<<16 | (uint32(rt)&15)<<12
+                o2 |= (uint32(rt)&15)<<16 | (uint32(rt)&15)<<12
+                o1 |= y
+                o2 |= x
+
        case 3: /* add R<<[IR],[R],R */
                o1 = c.mov(p)
```

$X + 0x024ff230 = x + 0x02500000 - 0x00000dd0$

$X - 0x024ff230 = x - 0x02500000 + 0x00000dd0$

$0x024ff230 - X = 0x02500000 - X - 0x00000dd0$

FP Optimizations

`VNMUL` multiplies together two floating-point register values, and writes the negation of the result to the destination register.

Encoding T2/A2 VFPv2, VFPv3, VFPv4 (sz = 1 UNDEFINED in single-precision only variants)

```
VNMUL<c>.F64 <Dd>, <Dn>, <Dm>
VNMUL<c>.F32 <Sd>, <Sn>, <Sm>
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	0	D	1	0	Vn				Vd				1	0	1	sz	N	1	M	0	Vm			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cond				1	1	1	0	0	D	1	0	Vn				Vd				1	0	1	sz	N	1	M	0	Vm			

Vector Multiply Accumulate multiplies corresponding elements in two vectors, and accumulates the results into the elements of the destination vector.

Vector Multiply Subtract multiplies corresponding elements in two vectors, subtracts the products from corresponding elements of the destination vector, and places the results in the destination vector.

Encoding T2/A2 VFPv2, VFPv3, VFPv4 (sz = 1 UNDEFINED in single-precision only variants)

```
V<op><c>.F64 <Dd>, <Dn>, <Dm>
V<op><c>.F32 <Sd>, <Sn>, <Sm>
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	1	0	0	D	0	0	Vn				Vd				1	0	1	sz	N	op	M	0	Vm			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
cond				1	1	1	0	0	D	0	0	Vn				Vd				1	0	1	sz	N	op	M	0	Vm			

Before / After

```go
//go:noinline
func fs(q, w float32) float32 {
        return  -(q * w)
}
```

```
TEXT main.fs(SB) /home/pi/a.go
        a.go:7  0x8c1e4 ed9d0a01        VLDR [SP, #4], S0
        a.go:7  0x8c1e8 ed9d1a02        VLDR [SP, #8], S2
        a.go:7  0x8c1ec ee200a01        VMUL.F32 S2, S0, S0
        a.go:7  0x8c1f0 eeb10a40        VNEG.F32 S0, S0
        a.go:7  0x8c1f4 ed8d0a03        VSTR [SP, #12], S0
        a.go:7  0x8c1f8 e28ef000        ADD $0, R14, R15
```

```
TEXT main.fs(SB) /root/a.go
  a.go:6                0x96fb8                ed9d0a01               MOVF 0x4(R13), F0
  a.go:6                0x96fbc                ed9d1a02               MOVF 0x8(R13), F1
  a.go:7                0x96fc0                ee200a41               VNMUL.F32 S2, S0, S0
  a.go:7                0x96fc4                ed8d0a03               MOVF F0, 0xc(R13)
  a.go:7                0x96fc8                e28ef000               ADD $0, R14, R15
```

Benchmark Result

1.A special test case improved 12.6%.
https://github.com/benshi001/ugo1/blob/master/fpmul_test.go

```
name                    old time/op    new time/op    delta
FPMul-4                    398µs ± 1%     348µs ± 1%  -12.64%  (p=0.000 n=40+40)
```

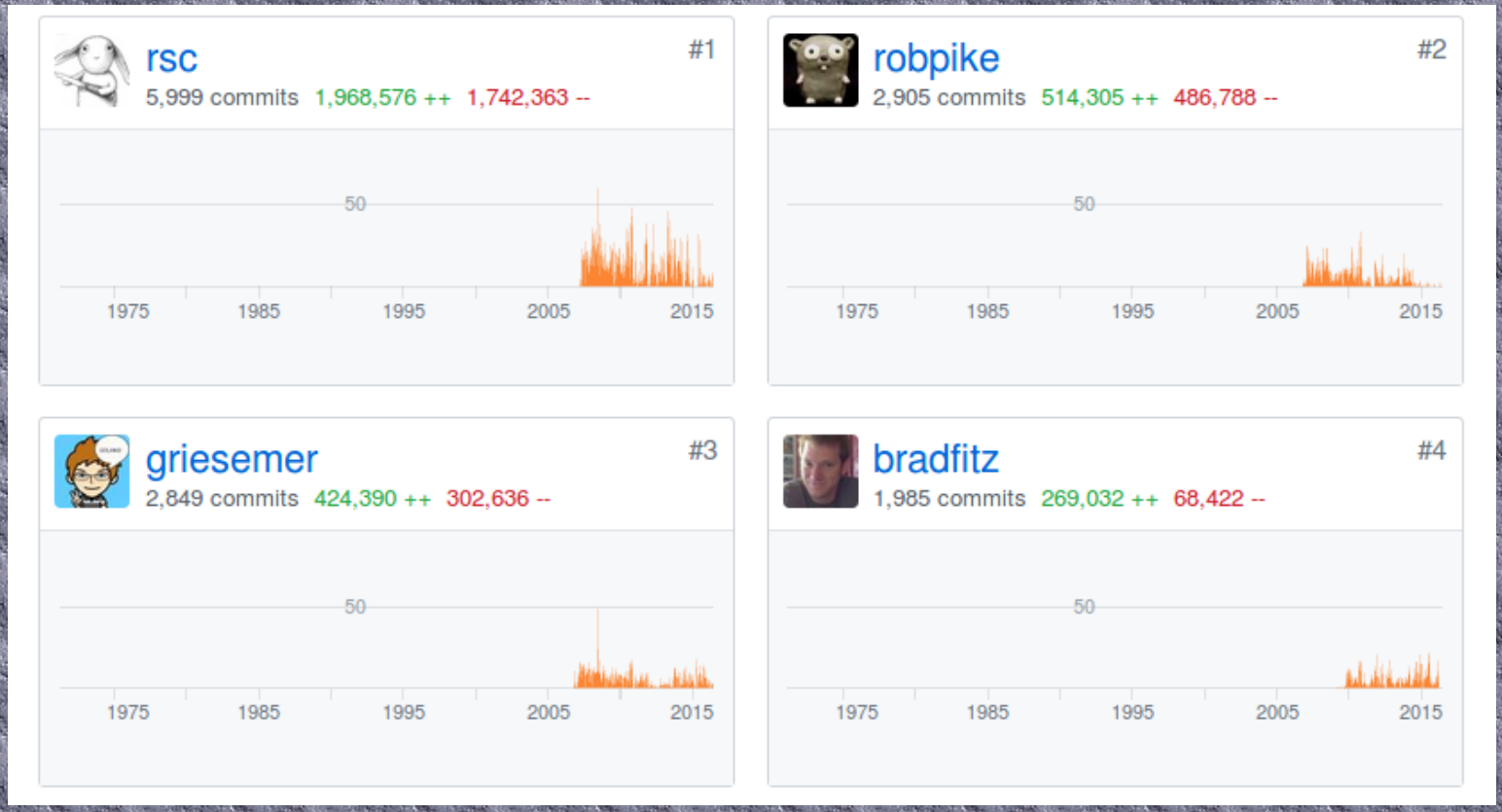
```
name                    old time/op    new time/op    delta
BinaryTree17-4             41.7s ± 1%     41.7s ± 1%     ~     (p=0.264 n=29+23)
Fannkuch11-4               24.2s ± 0%     24.1s ± 1%   -0.13%  (p=0.050 n=30+30)
FmtFprintfEmpty-4          826ns ± 1%     824ns ± 1%   -0.24%  (p=0.038 n=25+30)
FmtFprintfString-4        1.38µs ± 1%    1.38µs ± 0%   -0.42%  (p=0.000 n=27+25)
FmtFprintfInt-4           1.46µs ± 1%    1.46µs ± 0%     ~     (p=0.060 n=30+23)
FmtFprintfIntInt-4        2.11µs ± 1%    2.08µs ± 0%   -1.04%  (p=0.000 n=30+30)
FmtFprintfPrefixedInt-4   2.23µs ± 1%    2.22µs ± 1%   -0.51%  (p=0.000 n=30+30)
FmtFprintfFloat-4         4.49µs ± 1%    4.48µs ± 1%   -0.22%  (p=0.004 n=26+30)
FmtManyArgs-4             8.06µs ± 1%    8.12µs ± 1%   +0.68%  (p=0.000 n=25+30)
GobDecode-4                104ms ± 1%     104ms ± 2%     ~     (p=0.362 n=29+29)
GobEncode-4               92.9ms ± 1%    92.8ms ± 2%     ~     (p=0.786 n=30+30)
Gzip-4                     4.12s ± 1%     4.12s ± 1%     ~     (p=0.314 n=30+30)
Gunzip-4                   602ms ± 1%     603ms ± 1%     ~     (p=0.164 n=30+30)
HTTPClientServer-4         659µs ± 1%     655µs ± 2%   -0.64%  (p=0.006 n=25+28)
JSONEncode-4               234ms ± 1%     235ms ± 1%   +0.29%  (p=0.050 n=30+30)
JSONDecode-4               912ms ± 0%     911ms ± 0%     ~     (p=0.385 n=18+24)
Mandelbrot200-4           49.2ms ± 0%    41.7ms ± 0%  -15.35%  (p=0.000 n=25+27)
GoParse-4                 46.3ms ± 1%    46.3ms ± 2%     ~     (p=0.572 n=30+30)
```

Implemented via Peep Hole Rules

```
(NEGF (MULF x y)) && objabi.GOARM >= 6 -> (NMULF x y)
(NEGD (MULD x y)) && objabi.GOARM >= 6 -> (NMULD x y)
(MULF (NEGF x) y) && objabi.GOARM >= 6 -> (NMULF x y)
(MULD (NEGD x) y) && objabi.GOARM >= 6 -> (NMULD x y)
(NMULF (NEGF x) y) -> (MULF x y)
(NMULD (NEGD x) y) -> (MULD x y)
```

```
// the result will overwrite the addend, since they are in the same register
(ADDF a (MULF x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULAF a x y)
(ADDF a (NMULF x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULSF a x y)
(ADDD a (MULD x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULAD a x y)
(ADDD a (NMULD x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULSD a x y)
(SUBF a (MULF x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULSF a x y)
(SUBF a (NMULF x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULAF a x y)
(SUBD a (MULD x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULSD a x y)
(SUBD a (NMULD x y)) && a.Uses == 1 && objabi.GOARM >= 6 -> (MULAD a x y)
```

Contributor Ranking



Thank You!