معرفی کد میانی تسلنگ

در این مستند کد میانی تسلنگ (TSIR) معرفی می‌شود.

ساختار برنامه‌ها

هر فایل TSIR از یک یا بیشتر تابع تشکیل می‌شود. ماشین مجازی تسلنگ (TSVM) که کد میانی تسلنگ را اجرا می‌کند، اجرای یک فایل را از تابعی به نام main آغاز می‌نماید. در هر تابع مجموعه‌ای از رجیسترهای محلی در دسترس هستند. نام هر رجیستر از چسباندن حرف «r» با شماره‌ی رجیستر حاصل می‌شود. برای مثال r5 رجیستر پنجم یک تابع است. یک تابع با استفاده از کلمه‌ی کلیدی proc به شکل زیر تعریف می‌شود.

```
proc  main
   mov   r0, 0
   ret
```

یک تابع می‌تواند تعدادی ورودی دریافت نماید. در هنگام فراخوانی یک تابع با عملگر call، مقدار رجیسترهایی که بعد از نام تابع مشخص می‌شوند به ترتیب در رجیسترهای r0، r1 و ... تابع فراخوانی شده کپی می‌شوند. در هنگام برگشت از تابع با عملگر ret، مقداری که در تابع در رجیستر r0 قرار داده شده است در ورودی اول (اولین رجیستر بعد از نام تابع در عملگر call) کپی می‌شود. برای نمونه در تابع sum3 مجموع سه عدد محاسبه می‌گردد.

```
proc  sum3
   add   r0, r0, r1
   add   r0, r0, r2
   ret

proc  main
   call  iget, r3
   call  iget, r1
   call  iget, r2
   call  sum3, r3, r1, r2
   call  iput, r3
   mov   r0, 0
   ret
```

عملگرها

عملگرهای حسابی و مقایسه‌ای TSIR در جدول زیر نمایش داده می‌شوند.

دستور	توضیح
`mov r1, r2`	مقدار رجیستر **r2** را به **r1** می‌ریزد.
`mov r1, 5`	رجیستر **r1** را برابر عدد ۵ قرار می‌دهد.
`add r1, r2, r3`	حاصل جمع **r2** و **r3** را در **r1** قرار می‌دهد.
`sub r1, r2, r3`	حاصل تفریق **r3** از **r2** را در **r1** قرار می‌دهد.
`mul r1, r2, r3`	حاصل ضرب **r2** و **r3** را در **r1** قرار می‌دهد.
`div r1, r2, r3`	حاصل تقسیم **r2** بر **r3** را در **r1** قرار می‌دهد.
`mod r1, r2, r3`	باقیمانده‌ی تقسیم **r2** بر **r3** را در **r1** قرار می‌دهد.
`cmp= r1, r2, r3`	در صورتی که مقدار **r2** و **r3** برابر باشد، مقدار **r1** به یک و در غیر این صورت به صفر تغییر می‌کند.
`cmp< r1, r2, r3`	در صورتی که **r2** کوچک‌تر از **r3** باشد، مقدار **r1** به یک و در غیر این صورت به صفر تغییر می‌کند.
`cmp> r1, r2, r3`	در صورتی که **r2** بزرگ‌تر از **r3** باشد، مقدار **r1** به یک و در غیر این صورت به صفر تغییر می‌کند.
`cmp<= r1, r2, r3`	در صورتی که **r2** کوچک‌تر یا مساوی **r3** باشد، مقدار **r1** به یک و در غیر این صورت به صفر تغییر می‌کند.
`cmp>= r1, r2, r3`	در صورتی که **r2** بزرگ‌تر یا مساوی **r3** باشد، مقدار **r1** به یک و در غیر این صورت به صفر تغییر می‌کند.

عملگرهای پرش TSIR در جدول زیر نمایش داده می‌شوند.

دستور	توضیح
`jmp dst`	پرش به آدرس **dst** انجام می‌شود.
`jz r1, dst`	در صورتی که مقدار **r1** صفر باشد، پرش به آدرس **dst** انجام می‌شود.
`jnz r1, dst`	در صورتی که مقدار **r1** غیر صفر باشد، پرش به آدرس **dst** انجام می‌شود.

سایر عملگرهای TSIR در جدول زیر نمایش داده می‌شوند.

دستور	توضیح
`call func, r5, r9`	تابع **func** با ورودی‌های داده شده فراخوانی می‌شود.
`ret`	اجرای تابع خاتمه می‌یابد.
`ld r1, r2`	محتویات آدرسی که **r2** به آن اشاره می‌کند در **r1** ریخته می‌شود.
`st r1, r2`	مقدار **r1** در آدرسی که **r2** به آن اشاره می‌کند نوشته می‌شود.
`nop`	عملی انجام نمی‌دهد.

توابع داخلی

توابع داخلی TSVM در جدول زیر نمایش داده می‌شوند.

دستور	توضیح
`call iget, r1`	یک عدد از ورودی استاندارد می‌خواند و بر می‌گرداند.
`call iput, r1`	عدد داده شده را در خروجی استاندارد می‌نویسد.
`call mem, r1`	به مقدار خواسته شده حافظه تخصیص می‌دهد و آدرس حافظه را بر می‌گرداند.
`call rel, r1`	حافظه‌ی داده شده را آزاد می‌کند.

در نمونه‌ی زیر استفاده از توابع ورودی و خروجی نشان داده می‌شود.

```
proc  main
   call  iget, r0
   call  iput, r0
   mov   r0, 0
   ret
```

در نمونه‌ی زیر مدیریت و دسترسی به حافظه نشان داده می‌شود. در این نمونه، ۳۲ بایت حافظه تخصیص داده می‌شود و سپس آزاد می‌گردد.

```
proc  main
   mov   r1, 32
   call  mem, r1
   mov   r2, 5
   st    r2, r1            # write r2 to *r1
   ld    r3, r1            # read r3 from *r1
   call  rel, r1
   mov   r0, 0
   ret
```

نمونه‌های دیگر

در نمونه‌ی زیر استفاده از پرش شرطی نمایش داده می‌شود.

```
proc  main
   call  iget, r0
   call  iget, r1
   cmp<  r2, r0, r1
   jz    r2, out
   mov   r0, r1
out:
   call  iput, r0
   mov   r0, 0
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — آقای مشهدی

جواب اشتباه: ۰ از ۸ طول برنامه: ۱۱ متوسط زمان اجرا: ۳۰/۵

```
proc  help
   call  iget, r2
A:
   mod   r3, r0, r2
   mov   r0, r2
   mov   r2, r3
   jnz   r2, A
   ret
proc  main
   call  iget, r1
   call  help, r1
   call  help, r1
   call  iput, r1
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — خانم گلچین

جواب اشتباه: ۰ از ۸ طول برنامه: ۱۲ متوسط زمان اجرا: ۲۷/۴

```
proc  main
   call  iget, r0
   call  iget, r1
   call  iget, r2
   call  gcd2, r0, r1
   call  gcd2, r0, r2
   call  iput, r0
   ret
proc  gcd2
   mod   r2, r1, r0
   jz    r2, OUT
   call  gcd2, r2, r0
   mov   r0, r2
OUT:
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — خانم درجانی

جواب اشتباه: ۰ از ۸ طول برنامه: ۱۴ متوسط زمان اجرا: ۳۷/۸

```
proc  main
   call  iget, r1
Q:
   call  iget, r0
   cmp<  r2, r1, r0
   jz    r2, loop
   mov   r3, r1
m:
   mov   r1, r0
   mov   r0, r3
   jz    r3, s
loop:
   mod   r3, r1, r0
   jmp   m
s:
   jnz   r4, out
   mov   r4, 1
   jmp   Q
out:
   call  iput, r1
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — آقای یزدان‌پناه

جواب اشتباه: ۰ از ۸ طول برنامه: ۱۶ متوسط زمان اجرا: ۵۲/۴

```
proc  main
   call  iget, r0
   call  iget, r1
   call  iget, r2
   call  gcd, r0, r1
   call  gcd, r0, r2
   call  iput, r0
   ret
proc  gcd
begin:
   cmp>  r2, r1, r0
   jz    r2, do
   mov   r2, r0
   mov   r0, r1
   mov   r1, r2
do:
   jz    r1, end
   mod   r0, r0, r1
   jmp   begin
end:
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — خانم حقیقی

جواب اشتباه: ۰ از ۸ طول برنامه: ۱۸ متوسط زمان اجرا: ۳۰/۱

```
proc  main
   call  iget, r1
   call  iget, r2
   call  iget, r3
   call  gcd, r1, r2
   call  gcd, r1, r3
   call  iput, r1
   ret
proc  gcd
   cmp>  r4, r0, r1
   jz    r4, cnt
   mov   r5, r0
   mov   r0, r1
   mov   r1, r5
cnt:
loop:
   mod   r6, r1, r0
   jz    r6, out
   mov   r1, r0
   mov   r0, r6
   jmp   loop
out:
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — آقای مداحی

جواب اشتباه: ۰ از ۸ طول برنامه: ۲۱ متوسط زمان اجرا: ۶۳/۰

```
proc  main
   call  iget, r1
   call  iget, r2
   call  iget, r3
   mov   r0, 0
   mov   r4, 0
first:
   cmp== r5, r1, r0
   jz    r5, bnext
   mov   r1, r2
   jnz   r5, next
bnext:
   cmp== r5, r2, r0
   jnz   r5, next
   mov   r5, r2
   mod   r2, r1, r2
   mov   r1, r5
   jmp   first
next:
   jnz   r4, finish
   mov   r4, 1
   mov   r2, r3
   jmp   first
finish:
   call  iput, r1
   ret
```

محاسبه‌ی بزرگ‌ترین مقسوم علیه مشترک — آقای رایج

جواب اشتباه: ۳ از ۸ طول برنامه: ۳۴ متوسط زمان اجرا: ۴۳/۳

```
proc  main
   mov   r0, 8
   mov   r1, 8
   mov   r2, 8
   call  mem, r0
   call  mem, r1
   call  mem, r2
   call  iget, r3
   call  iget, r4
   call  iget, r5
   st    r3, r0
   st    r4, r1
   st    r5, r2
   call  findmin, r0, r1
   call  gcd, r0, r1
   call  findmin, r0, r2
   call  gcd, r0, r2
   ld    r6, r0
   call  iput, r6
   ret
proc  findmin
   ld    r2, r0
   ld    r3, r1
   cmp<= r4, r2, r3
   jnz   r4, out1
   st    r3, r0
   st    r2, r1
out1:
   ret
proc  gcd
   ld    r2, r0
   ld    r3, r1
op:
   mod   r4, r3, r2
   jz    r4, out2
   mov   r3, r2
   mov   r2, r4
out2:
   st    r2, r0
   ret
```