ریزپردازنده ۱

دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر

عبارات

عبارات

اسمبلر از عبارات ثابتی استفاده می‌کند. این عبارات می‌توانند از عملگر، عملوند، و توابع تشکیل شوند. تمام عبارات ذاتاً ۶۴ بیتی هستند.

عملوندها

عملوندهای زیر می‌توانند مورد استفاده قرار گیرند:

- برچسب‌های تعریف شده به وسیله کاربر که مقداری برابر با شمارنده‌ی مکان در محل ظهورشان دارند.
- متغیرهای تعریف شده به وسیله کاربر که با دستور SET تعریف شده‌اند.
- ثابت‌های تعریف شده به وسیله کاربر که با دستور EQU تعریف شده‌اند.

عملوندها

عملوندهای زیر می‌توانند مورد استفاده قرار گیرند:

- اعداد ثابت صحیح: ثابت‌ها به صورت‌های زیر می‌توانند آورده شوند:
  - مبنای ده (پیش‌فرض): 10, 255
  - مبنای شانزده (دارای دو نگارش): 0x0a, $0a, 0xff, $ff
  - مبنای دو: 0b00001010, 0b11111111
  - مبنای هشت: (با صفر شروع می‌شوند): 010, 077
- PC: مقدار کنونی شمارنده‌ی مکان در حافظه برنامه
- اعداد ثابت اعشاری

عملگرها

- اسمبلر از تعدادی عملگر پشتیبانی می‌کند که در اینجا توضیح داده شده‌اند. عملگری که زودتر ظاهر شود دارای اولویت بیشتری خواهد بود.
- عبارات می‌توانند در پرانتز قرار گیرند و به این صورت نسبت به عبارات مجاور پرانتز اولویت محاسبه داشته باشند.
- شرکت‌پذیری عملگرهای دودویی ترتیب محاسبه آنها در عبارات زنجیره‌ای را مشخص می‌کند.
  شرکت‌پذیری از چپ یعنی عملگر مربوطه اگر چند بار در یک عبارت ظاهر شود، شروع محاسبه از عملگر ظاهر شده در سمت چپ خواهد بود.
- بعضی عبارات شرکت‌پذیری ندارند. به این معنا که ظهور آنها در زنجیره بی‌معنی است.

عملگرها

عملگرهای زیر تعریف شده‌اند:

توصیف	نماد
کوچکتر	<
کوچکتر مساوری	<=
بزرگتر	>
بزرگتر مساوی	>=
مساوی	==
نابرابر	!=
AND بیتی	&
OR بیتی	|
XOR بیتی	^
AND منطقی	&&
OR منطقی	||
عملگر شرطی	?

توصیف	نماد
نقیض منطقی	!
نقیض بیتی	~
منفی یگانی (unary minus)	-
ضرب	*
تقسیم	/
باقیمانده	%
جمع	+
منها	-
شیفت به چپ	<<
شیفت به راست	>>

نقیض منطقی - !

عملگری یگانی که اگر عبارت ۰ باشد ۱ بر می‌گرداند و اگر عبارت غیر صفر باشد ۰ بر می‌گرداند.

اولویت : ۱۲

شرکت‌پذیری: ندارد

مثال:

```
Ldi r16, !0xF0 ; Load r16 with 0x00
```

نقیض بیتی - ~

عملگری یگانی که عبارت ورودی را به صورتی بر می‌گرداند که همه بیت‌ها در عبارت خروجی عکس شده‌اند.

اولویت : ۱۲

شرکت‌پذیری: ندارد

مثال:

```
Ldi r16, ~0xF0 ; Load r16 with 0x0F
```

منفی یگانی- -

عملگری یگانی که منفی عبارت مقابلش را برمی‌گرداند

اولویت : ۱۴

شرکت‌پذیری: ندارد

مثال:

```
Ldi r16, -2 ; Load r16 with -2 (0xFE)
```

ضرب - *

عملگری که ضرب دو عبارت را برمی‌گرداند.

اولویت : ۱۳

شرکت‌پذیری: از چپ

مثال:

```
Ldi r30, label*2 ; Load r30 with label*2
```

تقسیم - /

عملگری که خارج قسمت تقسیم عبارت سمت چپ بر عبارت سمت راست را بر می‌گرداند.

اولویت : ۱۳

شرکت‌پذیری: از چپ

مثال:

```
Ldi r30, label/2 ; Load r30 with label/2
```

باقیمانده - %

عملگری که باقیمانده تقسیم عبارت سمت چپ بر عبارت سمت راست را برمی‌گرداند.

اولویت : ۱۳

شرکت‌پذیری: از چپ

مثال:

```
Ldi r30, label%2 ; Load r30 with label%2
```

جمع - +

عملگری که مجموع دو عبارت را برمی‌گرداند.

اولویت : ۱۲

شرکت‌پذیری: از چپ

مثال:

```
Ldi r30, c1+c2 ; Load r30 with c1+c2
```

تفریق - -

عملگری که عبارت سمت راست را از عبارت سمت چپ می‌کاهد و برمی‌گرداند.

اولویت : ۱۲

شرکت‌پذیری: از چپ

مثال:

```
Ldi r17, c1-c2 ; Load r17 with c1-c2
```

شیفت به چپ - <<

عملگری دوگانی که عبارت سمت چپ را به تعداد عبارت سمت راست شیفت به چپ می‌دهد و برمی‌گرداند.

اولویت : ۱۱

شرکت‌پذیری: از چپ

مثال:

```
Ldi r17, 1<<bitmask ; Load r17 with 1 shifted left bitmask times
```

شیفت به راست - >>

عملگری دوگانی که عبارت سمت چپ را به تعداد عبارت سمت راست به سمت راست شیفت می‌دهد و برمی‌گرداند.

اولویت : ۱۱

شرکت‌پذیری: از چپ

مثال:

```
Ldi r17, c1>>c2 ; Load r17 with c1 shifted right c2 times
```

کوچکتر - <

عملگری دوگانی که اگر عبارت علامت‌دار سمت چپ از عبارت علامت‌دار سمت راست کوچکتر باشد ۱ برمی‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۱۰

شرکت‌پذیری: ندارد

مثال:

```
ori r18,bitmask*(c1<c2)+1 ;Or r18 with an expression
```

کوچکتر یا مساوی - =>

عملگری دوگانی که اگر عبارت علامت‌دار سمت چپ از عبارت علامت‌دار سمت راست کوچکتر یا با آن مساوی باشد ۱ برمی‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۱۰

شرکت‌پذیری: ندارد

مثال:

```
ori r18,bitmask*(c1<=c2)+1 ;Or r18 with an expression
```

بزرگتر- >

عملگری دوگانی که اگر عبارت علامت‌دار سمت چپ از عبارت علامت‌دار سمت راست بزرگتر باشد ۱ برمی‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۱۰

شرکت‌پذیری: ندارد

مثال:

```
ori r18,bitmask*(c1>c2)+1 ;Or r18 with an expression
```

بزرگتر یا مساوی- =>

عملگری دوگانی که اگر عبارت علامت‌دار سمت چپ از عبارت علامت‌دار سمت راست بزرگتر یا با آن مساوی باشد ۱ برمی‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۱۰

شرکت‌پذیری: ندارد

مثال:

```
ori r18,bitmask*(c1>=c2)+1 ;Or r18 with an expression
```

مساوی- ==

عملگری دوگانی که اگر عبارت علامت‌دار سمت چپ با عبارت علامت‌دار سمت راست مساوی باشد ۱ برمی‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۹

شرکت‌پذیری: ندارد

مثال:

```
andi r19,bitmask*(c1==c2)+1 ;And r19 with an expression
```

AND بیتی - &

عملگری که نتیجه And بیت‌های متناظر دو عبارت را برمی‌گرداند.

اولویت : ۸

شرکت‌پذیری: از چپ

مثال:

```
ldi r18,High(c1&c2) ;Load r18 with an expression
```

XOR بیتی - ^

عملگری که نتیجه Xor بیت‌های متناظر دو عبارت را برمی‌گرداند.

اولویت : ۷

شرکت‌پذیری: از چپ

مثال:

```
ldi r18,Low(c1^c2) ;Load r18 with an expression
```

OR بیتی - |

عملگری که نتیجه or بیت‌های متناظر دو عبارت را برمی‌گرداند.

اولویت : ۶

شرکت‌پذیری: از چپ

مثال:

```
ldi r18,Low(c1|c2) ;Load r18 with an expression
```

AND منطقی - &&

عملگری که در صورت غیر صفر بودن هر دو عبارت ۱ برمی‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۵

شرکت‌پذیری: از چپ

مثال:

```
ldi r18,Low(c1&&c2) ;Load r18 with an expression
```

OR منطقی - ||

عملگری که در صورت غیر صفر بودن حداقل یکی از عبارات، ۱ بر می‌گرداند و در غیر این صورت صفر برمی‌گرداند.

اولویت : ۴

شرکت‌پذیری: از چپ

مثال:

```
ldi r18,Low(c1||c2) ;Load r18 with an expression
```

عملگر شرطی - : ؟

سینتکس استفاده: condition? expression1 : expression2

عملگری سه‌گانی که در صورت برقرار بودن شرط condition عبارت expression1 و در غیر این صورت expression2 را برمی‌گرداند.

اولویت : ۳

شرکت‌پذیری: ندارد

مثال:

```
ldi r18, a > b? a : b ; Load r18 with the maximum numeric value of a and b
```

توجه! این قابلیت در AVRASM 2.1 معرفی شده و در نسخه‌های قبلی موجود نیست.

توابع

- Low(expression) بایت کم ارزش یک عبارت را برمی‌گرداند.
- High(expression) بایت دوم یک عبارت را برمی‌گرداند.
- Byte2(expression) مشابه تابع high است.
- Byte3(expression) بایت سوم یک عبارت را برمی‌گرداند.
- Byte4(expression) بایت چهارم یک عبارت را برمی‌گرداند.
- Lwrd(expression) بیت‌های ۰ تا ۱۵ یک عبارت (کلمه کم ارزش عبارت) را برمی‌گرداند.
- Hwrd(expression) بیت‌های ۱۶ تا ۳۱ یک عبارت را برمی‌گرداند.

توابع

- Page(expression) بیت‌های ۱۶ تا ۲۱ یک عبارت را برمی‌گرداند.
- Exp2(expression) عدد دو را به توان عبارت داده شده می‌رساند و برمی‌گرداند.
- Log2(expression) بخش صحیح لگاریتم در پایه دو عبارت داده شده را برمی‌گرداند.
- Int(expression) یک عدد اعشاری ممیز شناور را به یک عدد صحیح می‌کاهد (بخش بعد از ممیز را حذف می‌کند)
- Frac(expression) بخش بعد از ممیز عدد اعشاری ممیز شناور را جدا می‌کند (بخش صحیح را حذف می‌کند)

توابع

- Q7(expression) عدد اعشاری ممیز شناور را به شکلی مناسب برای دستورات Fmul, Fmuls, Fmulsu در می‌آورد (علامت + ۷ بیت اعشار)
- Q15(expression) عدد اعشاری ممیز شناور را به شکلی مناسب برای دستورات Fmul, Fmuls, Fmulsu در می‌آورد (علامت + ۱۵ بیت اعشار)
- Abs(expression) قدرمطلق عبارت ثابت داده شده را برمی‌گرداند.

توابع

- Defined(symbol) این تابع true برمی‌گرداند اگر نماد symbol قبلاً به وسیله دستوراتی مانند .equ, .set, .def تعریف شده باشد. معمولاً در کنار if. استفاده می‌شود (if defined(foo)) اما در هر زمینه‌ای می‌تواند به کار رود. تفاوتش با سایر توابع این است که گذاشتن پرانتز جلوی آن الزامی نیست.
- Strlen(string) طول یک رشته ثابت را برمی‌گرداند