ریزپردازنده ۱

دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر

مشکلات شناخته‌شده در اسمبلر AVR

سرفصل مطالب

- ادامه دادن خط در فراخوانی ماکروها امکان‌پذیر نیست
- جا انداختن خط جدید در انتهای فایل
- عملگرهای کاهش و افزایش
- رجوع به آدرس‌های بعدی در گزاره‌های شرطی
- پیغام‌های خطا
- با Defined اشتباهاً به عنوان کلمه کلیدی اسمبلربرخورد می‌شود
- مسائل پیش‌پردازنده‌ها

ادامه دادن خط در فراخوانی ماکروها امکان‌پذیر نیست

برنامه زیر این مشکل را نشان می‌دهد

```
.macro m
Ldi @0, @1
.endm m r16, \ 0
```

این مشکل ناشی از ماکروهای پیش‌پردازنده‌ها نیست. (#define)

جا انداختن خط جدید در انتهای فایل

- AVRASM2 در صورتی که در انتهای فایل سورس خط جدید(newline) وجود نداشته باشد به مشکلاتی برمی‌خورد.
- اگر خطایی در خط آخر یک فایل پیوست شده (include) وجود داشته باشد با پیغام‌های خطایی از جمله اشتباه بودن نام فایل یا شماره خط یا در مواردی خطای سینتکس مواجه خواهیم شد.
- توجه: ویرایشگر Atmel Studio به طور خودکار در انتهای فایل سورس خط جدید اضافه نمی‌کند.

عملگرهای کاهش و افزایش

- عملگرهای کاهش یا افزایش (--/++) توسط اسمبلر شناخته می‌شوند اما ممکن است باعث بروز رفتاری غیرمترقبه شوند.
- به عنوان مثال 1-- باعث بروز خطای سینتکس می‌شود و در صورتی که منظور از این عبارت (1-)- است باید از 1- - استفاده کرد.
- عملگرهای ++/-- در اسمبلرهای کنونی استفاده‌ای ندارند، اما برای استفاده‌های آینده رزرو شده‌اند.

رجوع به آدرس‌های بعدی در گزاره‌های شرطی

- رجوع دادن به خطوط بعدی در گزاره‌های شرطی ممکن است باعث بروز نتایج غیرمترقبه شود و در بعضی موارد حتی ممکن است غیرمجاز باشد. به عنوان مثال:

```
.org LARGEBOOTSTART
; the following sets up RAMPZ:Z to point to a FLASH data object, typically
; for use with ELPM.

ldi ZL, low (cmdtable * 2)

ldi ZH, high (cmdtable * 2)
.if ((cmdtable * 2) > 65535)

ldi r16, 1

sts RAMPZ, r16
.endif

; more code follows here
cmdtable: .db "foo", 0x0
```

رجوع به آدرس‌های بعدی در گزاره‌های شرطی

دلیل بروز چنین مشکلی این است که نتیجه گزاره شرطی ممکن است مقدار برچسب خط رجوع داده شده در جلوتر را تغییر دهد و این تغییر باعث تغییر روند برنامه شود.

چنین برنامه‌ای مجاز است:

```
.ifdef FOO
nop ; some code here
.endif
rjmp label ; more code here
.equ FOO = 100
label: nop
```

در این مثال FOO در هنگامی که در گزاره شرطی استفاده شده تعریف نشده است. استفاده از **ifdef** در اینجا مجاز بوده و نتیجه گزاره شرطی **false** است. اما استفاده از برنامه فوق توصیه نمی‌شود زیرا هدف برنامه‌نویس مشخص نمی‌شود. بهتر از برنامه صفحه بعد استفاده شود

رجوع به آدرس‌های بعدی در گزاره‌های شرطی

```
; Define FOO if it is not already defined.
.ifndef FOO
.equ FOO = 0x100
.endif
```

توجه: در گزاره‌های شرطی پیش‌پردازنده (#if/#ifdef) این موقعیت‌ها خوش تعریف هستند، علائم پیش‌پردازنده تا زمان رسیدن به تعریف آن‌ها، تعریف نشده هستند و چنین خطاهایی هرگز بروز پیدا نخواهند کرد.

پیغام‌های خطا

گاهی فهم پیغام‌های خطا دشوار می‌شود. معمولاً اشتباه نگارشی در بعضی موارد پیغامی مانند زیر تولید می‌کند:

myfile.asm(30): error: syntax error, unexpected FOO

که در آن FOO کلمه‌ای نامفهوم است که دارای اشتباه نگارشی است. اما نام فایل و خط رجوع داده شده صحیح هستند.

با DEFINED اشتباهاً به عنوان کلمه کلیدی اسمبلربرخورد می‌شود

کلمه کلیدی DEFINED(symbol) در تمام زمینه‌ها شناخته می‌شود در صورتی که باید تنها در گزاره‌های شرطی شناخته شود.

این مسئله مانع از این می‌شود که بتوان از DEFINED(symbol) به عنوان علائم کاربر از جمله برچسب‌ها و غیره استفاده کرد.

از طرفی اجازه می‌دهد که ساختارهایی مانند ‘dw foo = defined(bar).’ داشته باشیم که نباید ممکن باشد.

باید توجه داشت که پیش‌پردازنده و اسمبلر پیاده‌سازی‌های مختلفی از DEFINED(symbol) دارند.

با DEFINED اشتباهاً به عنوان کلمه کلیدی اسمبلربرخورد می‌شود

رفتار دقیق DEFINED(symbol) اکنون در نسخه ۲.۱.۵ به صورت زیر است:

- کلمه کلیدی 'defined' برای پیش‌پردازنده فقط با علائمی که با #define تعریف شده‌اند مرتبط می‌شود، و اکنون این کار را تنها برای گزاره‌های شرطی پیش‌پردازنده (#if/#elif) انجام می‌دهد.
- در کد باقیمانده دید اسمبلر از DEFINED(symbol) به کار گرفته می‌شود و رفتار صحیح این خواهد بود که فقط در گزاره‌های شرطی اسمبلر (.if/.elif) شناخته شود.

مسائل پیش‌پردازنده‌ها

- پیش‌پردازنده، دایرکتیو های نادرست پیش‌پردازنده را درون یک گزاره شرطی false تشخیص نمی‌دهد. چنین مسئله‌ای می‌تواند به اشتباهات نگارشی مانند زیر بیانجامد:

```
#if __ATmega8__
//...
#elseif __ATmega16__ //WRONG, the correct directive is #elif
// This will go undetected if __ATmega8__ is false
//...
#else
// when __ATmega8__ is false this section will be assembled even if
// __ATmega16__ is true.
#endif
```

اینکه پیشامد فوق یک باگ به حساب می‌آید یا نه قابل بحث است. رفتار فوق مطابق با پیش‌پردازنده زبان C هست.

مسائل پیش‌پردازنده‌ها

- مشکل شماره ۳۳۶۱: پیش‌پردازنده اشتباهاً اجازه اضافه کردن متن پس از دایرکتیو ها را می‌دهد که ممکن است باعث پیشامدهای غیرمترقبه شود. به عنوان مثال #endif #endif بدون ایجاد پیغام خطا یا هشدار به عنوان یک #endig واحد شناخته می‌شود.

مسائل پیش‌پردازنده‌ها

- مشکل شماره ۴۷۴۱: گزاره‌های شرطی اسمبلر در ماکروهای پیش‌پردازنده‌ها کار نمی‌کنند. استفاده از ماکرو تعریف شده در زیر بر حسب نتیجه گزاره شرطی (True یا False) باعث ایجاد خطای syntax می‌شود.

```
#define TEST \
.IF val \
.DW 0 \
.ELSE \
.DW 1 \
.ENDIF
```

دلیل این مسئله آن است که گزاره‌های شرطی اسمبلر باید در خطی جداگانه قرار داشته باشند، و ماکروی پیش‌پردازنده مانند بالا در یک خط متمرکز شده.