ریزپردازنده ۱

دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر

دایرکتیوهای اسمبلر

دایرکتیو چیست؟

- دایرکتیو ها شبه دستوراتی هستند که اسمبلر را برای انجام کاری هدایت می کنند. به طور مثال با استفاده از دایرکتیو ها می توان
  - آدرس شروع برنامه را به اسمبلر اعلام نمود.
  - محلی از حافظه را برای یک متغیر تخصیص داد و با استفاده از یک **label** به آن اشاره نمود.
  - یک فایل دیگر را به کدی که اسمبلر در حال اجرا است را اضافه **(include)** کرد.
- دایرکتیو ها با نقطه شروع می شوند

دایرکتیو DEF

- یک اسم نمادین (symbolic) به یک ثبات اختصاص می دهد که باعث راحتی در برنامه نویسی می شود. هر ثبات می تواند چندین اسم نمادین داشته باشد

.DEF Symbol=register

```
.DEF temp=R16
.DEF ior=R0

ldi temp,0xf0 ; Load 0xf0 into temp register
in ior,0x3f ; Read SREG into ior register
eor temp,ior ; Exclusive or temp and ior
```

دایرکتیو UNDEF

- عدم تخصیص اسم نمادین (symbolic) از یک ثبات.
- با این کار می توان از هر ثبات در محدوده scope های مختلفی بدون دریافت warning استفاده کرد.

.UNDEF Symbol

```
.DEF var1 = R16
ldi var1, 0x20
... ; do something more with var1
.UNDEF var1

.DEF var2 = R16 ; R16 can now be reused without warning.
```

دایرکتیو BYTE

- با استفاده از این دایرکتیو می توان محلی از حافظه SRAM یا EEPROM را رزرو کرد. با استفاده از یک label همراه دایرکتیو، می توان به محل رزرو شده دسترسی پیدا کرد. دایرکتیو BYTE یک پارامتر دارد که تعداد بایت مورد نیاز برای رزرو را مشخص می کند.
- (نکته: از دایرکتیو BYTE نمی توان داخل سگمنت کد استفاده کرد)

Label : .BYTE expression

```
var1:   .BYTE 1 ; reserve 1 byte to var1
table: .BYTE tab_size ; reserve tab_size bytes
```

```
ldi r30,low(var1) ; Load Z register low
ldi r31,high(var1) ; Load Z register high
```

دایرکتیو DB و DW و DD و DQ

- دایرکتیو های DB، DW، DD و DQ به ترتیب برای تعریف byte، word، doubleword و quadword **ثابت** در حافظه برنامه یا EEPROM به کار می روند. برخلاف دایرکتیو BYTE، این دایرکتیو ها می توانند لیستی از مقادیر به عنوان پارامتر بگیرند.

```
consts: .DB 0, 255, 0b01010101, -128, 0xaa
varlist: .DW 0, 0xffff, 0b1001110001010101, -32768, 65535
varlist: .DD 0, 0xfadebabe, -2147483648, 1 << 30
eevarlst: .DQ 0,0xfadebabedeadbeef, 1 << 62
```

دایرکتیو سگمنت CSEG

- این دایرکتیو شروع سگمنت code را مشخص می کند. در فایل می توان چندین بخش از این دایرکتیو های CSEG داشت که در زمان اسمبلی همه آن ها با هم تجمیع می شوند. دایرکتیو BYTE را نمی توان در سگمنت کد استفاده نمود. همچنین این سگمنت پیش فرض اسمبلی است.

```
.CSEG
ldi r30,low(var1) ; Load Z register low
ldi r31,high(var1) ; Load Z register high
ld r1,Z ; Load VAR1 into register 1
```

دایرکتیو CSEGSIZE

- با استفاده از این دایرکتیو می توان اندازه حافظه برنامه را تعیین نمود. حافظه برنامه و حافظه داده در SRAM به سه بخش تقسیم می شود:
  - 10K *16 بیت مخصوص حافظه برنامه
  - 4K*8 بیت مخصوص حافظه داده
  - 6k*16 یا 12k*8 بیت نیز می توان به حافظه برنامه یا داده در بلوک های 2k*16 یا 4k*8 اختصاص داد.

.CSEGSIZE = 10 | 12 | 14 | 16

```
.CSEGSIZE = 12 ; Specifies the program meory size as 12K x 16
```

دایرکتیو سگمنت DSEG

- این دایرکتیو شروع سگمنت data را مشخص می کند. در فایل می توان چندین بخش از این دایکتیو های DSEG داشت که در زمان اسمبلی همه آن ها با هم تجمیع می شوند. در این دایرکتیو معمولا تعریف های label ها و دایرکتیو های BYTE گنجانده می شود.

```
.DSEG ; Start data segment
var1: .BYTE 1 ; reserve 1 byte to var1
table: .BYTE tab_size ; reserve tab_size bytes.

.CSEG
ldi r30,low(var1) ; Load Z register low
ldi r31,high(var1) ; Load Z register high
ld r1,Z ; Load var1 into register 1
```

دایرکتیو سگمنت ESEG

- این دایرکتیو شروع یک سگمنت در حافظه EEPROM را مشخص می کند. در فایل می توان چندین بخش از این دایکتیو های ESEG داشت که در زمان اسمبلی همه آن ها با هم تجمیع می شوند. در این دایرکتیو معمولا تعریف های label ها و دایرکتیو های DB و DW گنجانده می شود.

```
.DSEG ; Start data segment
var1: .BYTE 1 ; reserve 1 byte to var1
table: .BYTE tab_size ; reserve tab_size bytes.

.ESEG
eevar1: .DW 0xffff ; initialize 1 word in EEPROM
```

دایرکتیو EQU

- این دایرکتیو یک مقدار را به یک label اختصاص می دهد که در طول برنامه نمی توان آن را تغییر داد.

.EQU label=expression

```
.EQU io_offset = 0x23
.EQU porta = io_offset + 2.CSEG ; Start code segment
clr r2 ; Clear register 2
out porta,r2 ; Write to Port A
```

دایرکتیو SET

- این دایرکتیو یک مقدار را به یک label اختصاص می دهد که بر خلاف دایرکتیو EQU در طول برنامه می توان مقدار آن را تغییر داد.

.SET label=expression

```
.SET FOO = 0x114; set FOO to point to an SRAM location
lds r0, FOO; load location into r0
.SET FOO = FOO + 1 ; increment (redefine) FOO. This would be illegal if using .EQU
lds r1, FOO ; load next location into r1
```

دایرکتیو EXIT

- در حالت کلی اسمبلی یک فایل را تا رسیدن به انتهای فایل EOF می خواند. استفاده از دایرکتیو EXIT باعث می شود که پیش از رسیدن به EOF از فایل خارج می شود.

.EXIT

```
.EXIT ; Exit this file
```

دایرکتیو INCLUDE

- این دایرکتیو اسمبلی را به خواندن یک فایل دیگر هدایت می کند تا زمانی که آن فایل به انتها برسد و یا اسمبلی دایرکتیو EXIT را در آن فایل ببیند.

.INCLUDE “filename”

```
; iodefs.asm:
.EQU sreg = 0x3f ; Status register
.EQU sphigh = 0x3e ; Stack pointer high
.EQU splow = 0x3d ; Stack pointer low

; incdemo.asm
.INCLUDE iodefs.asm ; Include I/O definitions
in r0,sreg ; Read status register
```

دایرکتیو MICRO, ENDMIRCO

- این دایرکتیو که همراه با یک اسم می اید، به اسمبلی شروع یک مایکرو را اعلام می کند. در طول برنامه هر جا که اسم مایکرو (به همراه یک سری پارامتر) نوشته شده باشد، تعریف آن مایکرو در آن جا قرار می گیرد.
- در داخل هر مایکرو می توان تا ۱۰ پارامتر استفاده کرد که آن ها را با @0-@9 نشان می دهند.
- ENDMICRO نیز پایان مایکرو را اعلام می کند

```
.MACRO SUBI16 ; Start macro definition
subi @1,low(@0) ; Subtract low byte
sbci @2,high(@0) ; Subtract high byte
.ENDMACRO ; End macro definition

.CSEG ; Start code segment
SUBI16 0x1234,r16,r17 ; Sub.0x1234 from r17:r16
```

دایرکتیوهای شرطی IF, ELIF, ELSE, ENDIF

- در زبان اسمبلی می توان جریان اجرای دستورات را با استفاده از دایرکتیو های شرطی کنترل کرد.
اگرexpression مخالف صفر شود دستورات بعد از دایرکتیو IF تا زمان رسیدن به دایرکتیو ELSE و یا ELIF اجرا می شوند.

```
.IF <expression>

...

.ELSE | .ELIF<expression>

...

.ENDIF
```

دایرکتیوهای شرطی IFDEF, IFNDEF

- در دایرکتیو IFDEF اگر اسمبلی با استفاده از EQU یا SET تعریف شده باشد ، دستورات بعد از این دایرکتیو تا رسیدن به دایرکتیو ENDIF و یا ELSE اجرا می کند.
- دایرکتیو IFNDEF عکس شرط IFDEF را دارد.

```
.IFDEF <symbol> |.IFNDEF <symbol>
...
.ELSE | .ELIF<expression>
...
.ENDIF
```

دایرکتیو ERROR, WARNING, MESSAGE

```
.IFDEF DEBUG
.MESSAGE "Debug mode"
.ENDIF
```

- دایرکتیو Message یک string به عنوان خروجی می دهد.

- دایرکتیو Warning نیز مانند Message یک string به عنوان خروجی می دهد

```
.IFDEF EXPERIMENTAL_FEATURE
.WARNING "This is not properly tested, use at own risk."
.ENDIF
```

- دایرکتیو Error یک string خروجی می دهد و برنامه را متوقف می کند

```
.IFDEF TOBEDONE
.ERROR "Still stuff to be done.."
.ENDIF
```

دایرکتیو LIST, NOLIST

- دایرکتیو LIST به اسمبلر اعلام می کند که تولید LIST FILE را فعال کند. (حالت پیش فرض فعال است). با استفاده از دایرکتیو NOLIST می توان برای قسمت های انتخاب شده از کد LIST FILE تولید کرد.

```
.NOLIST ; Disable listfile generation
.INCLUDE "macro.inc" ; The included files will not
.INCLUDE "const.def" ; be shown in the listfile
.LIST ; Reenable listfile generation
```

دایرکتیو LISTMAC

- در حالت کلی تنها فراخوانی مایکرو ها در LIST FILE قرار می گیرند. اگر بخواهیم تعریف مایکرو ها نیز دی LIST FILE قرار گیرند باید از دایرکتیو LISTMAC استفاده کنیم.

```
.MACRO MACX ; Define an example macro
add r0,@0 ; Do something
eor r1,@1 ; Do something
.ENDMACRO ; End macro definition

.LISTMAC ; Enable macro expansion

MACX r2,r1 ; Call macro, show expansion
```

دایرکتیو ORG

- دایرکتیو ORG شمارنده آدرس را مقداردهی می کند. اگر در سگمنت دیتا باشیم، شمارنده آدرس SRAM مقداردهی می شود. اگر در سگمنت کد باشیم، شمارنده آدرس حافظه برنامه مقدار دهی می شود و اگر در سگمنت EEPROM باشیم، شمارنده آدرس حافظه EEPROM مقدار دهی می شود.

```
.DSEG ; Start data segment
.ORG 0x120; Set SRAM address to hex 120
variable: .BYTE 1 ; Reserve a byte at SRAM adr. 0x120

.CSEG
.ORG 0x10 ; Set Program Counter to hex 10
mov r0,r1 ; Do something
```

دایرکتیو OVERLAP/NOOVERLAP

- با استفاده از دایرکتیو overlap و nooverlap می توان بازه ای از کد را مشخص کرد که بعدا بتوان توسط یک کد دیگر (بدون خطا) جایگزین گردد.

```
.overlap
.org 0 ; section #1
rjmp default
.nooverlap
.org 0 ; section #2
rjmp RESET ; No error given here
.org 0 ; section #3
rjmp RESET ; Error here because overlap with #2
```