

راهنمای کار با شبیه‌ساز IBM360

محیط برنامه

- برای مدیریت فایل متن کد خود (با فرمت .txt) از منوی file استفاده کنید.

- بصورت خودکار با خروج از برنامه، کد در فایل به نام temporary.txt ذخیره می‌شود. با باز کردن

دوباره‌ی برنامه، این کد بصورت خودکار لود می‌شود.

- میانبرهای معمول ویرایشی و مدیریت فایل قابل استفاده اند.

- با زدن دکمه‌ی show/hide machine code کد اسمبلی شما که به زبان ماشین ترجمه شده است،

در سمت راست صفحه نمایش داده خواهد شد؛ ترجمه بصورت لحظه‌ای و بعد از هر تغییر در کد نمایش داده می‌شود.

- خطی از کد که نشانه‌گر روی آن است، همچنین آدرس و کد ماشین آن خط، زردرنگ می‌شود (شکل ۱).

show/hide machine code	Execute		Address	Machine Code
1 <i>Sum start</i> 0		; segment declaration		
2 <i>STM R14, R12, 12(R13)</i>		; storing the original values of registers	00000	90ECD00C
3 <i>BALR R12, R0</i>		; storing PC value in R12	00000	05C0
4 <i>using *, R12</i>		; declaration of R12 as the base register		
5				
6 <i>ST R13, Regl3</i>		; storing R13 in Regl3	00000	50D0C0AA
7 <i>LA R13, RegSaveArea</i>		; R13 <- address of RegSaveArea, this will be passed to the child segment	00000	41D0C072
8				
9 <i>LA R1, Var</i>		; R1 <- address of Var, this will be passed to the child segment	00000	4110C03A
10 <i>LA R2, P</i>		; R2 <- address of P, this will be passed to the child segment	00001	4120C06A
11 <i>LA R3, 12</i>		; n <- 12 (R3 will be passed to the child segment)	00001	4130000C
12 <i>L R15, =V(Product)</i>		; R15 <- address of the beginning of the child segment	00001	58F0C0AE
13 <i>XR R4, R4</i>		; R4 <- 0, R4 will hold the sum of the products	00001	1744
14				
15 <i>loop: EM out</i>		; break out of the loop if n < 0	00002	4740C02C
16 <i>BALR R14, R15</i>		; branch to the child segment, with R14 holding the return address	00002	05EF
17 <i>A R4, P</i>		; add the product calculated by the child segment to R14	00002	5A40C06A
18 <i>S R3, =F'4'</i>		; n <- n - 4	00002	5B30C0B2
19 <i>B loop</i>			00002	47F0C01A
20				
21 <i>out: ST R4, Result</i>		; Store the final result	00003	5040C06E
22				
23 <i>L R13, Regl3</i>		; restoring R13 from Regl3	00003	58D0C0AA
24 <i>LM R14, R12, 12(R13)</i>		; restoring the original values of registers	00003	98ECD00C
25 <i>BR R14</i>		; returning control to OS	00003	07FE
26			00004	000000030
27 <i>Var DC F '3, 1, 2, 0, 4, -1, 5, 2, 6, -2, 1, 3'</i>		; a 4x3 matrix with arbitrary element values Var[i, j] for i in {0, 1, 2, 3} and j in {0, 1, 2}		
28		; note that Var[i, j] = *(Var + n x 3 + m) for n in {0, 4, 8, 12} and m in {0, 4, 8} since Var has		
29 <i>P DS F</i>		; P = Product(Var[i, j]) over j in {0, 1, 2} for i in each iteration	00007	00000000
30 <i>Result DS F</i>		; Result = Sum(Product(Var[i, j]) over j in {0, 1, 2}) over i in {0, 1, 2, 3}	00007	00000000
31 <i>RegSaveArea DS 14 F</i>		; variable to store the values of the registers by the child segment	00007	000000000
32 <i>Regl3 DS F</i>		; variable to store the value of R13 so that we can restore it after the child segment has changed	00008	00000000
33				
34 <i>end</i>		; end of segment		
35				
36				
37 <i>Product start</i> 0		; segment declaration		
38 <i>STM R14, R12, 0(R13)</i>		; storing the register values of the parent segment in its RegSaveArea variable	00000	90ECD000
39 <i>BALR R12, R0</i>		; storing PC value in R12	00000	05C0
40 <i>using *, R12</i>		; declaration of R12 as the base register		
41				
42		; R1 = address of Var, R2 = Address of P, R3 = n, and we need these values in this segment		
43			00000	41900001
44 <i>LA R9, 1</i>		; R9 <- 1, R9 will hold the product of the elements in the current row	00000	41500008
45 <i>LA R5, 8</i>		; m <- 8		
46			00000	4740C022
47 <i>loop: EM out</i>		; break out of the loop if m < 0	00001	41700003
48 <i>LA R7, 3</i>			00001	1C63
49 <i>MR R6, R3</i>		; [R6, R7] <- n x 3, note that R7 holds the value since the multiplied numbers were small	00001	1A75
50 <i>AR R7, R5</i>		; R7 <- n x 3 + j	00001	1A71
51			00001	5B30C0B2

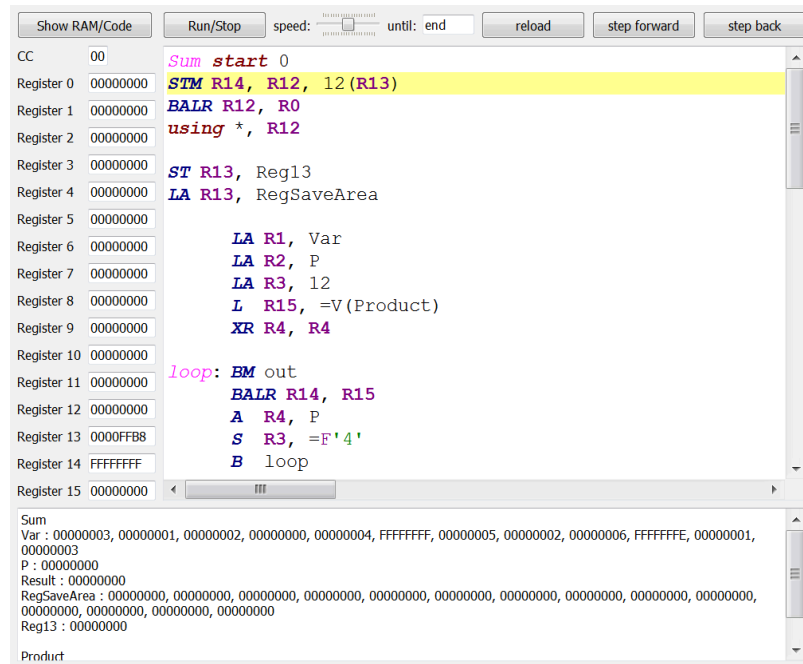
شکل ۱. ویرایشگر کد اسمبلی و نمایشگر کد ماشین.

- با زدن دکمه‌ی execute، صفحه‌ای برای اجرای کد باز می‌شود (شکل ۲.۱).

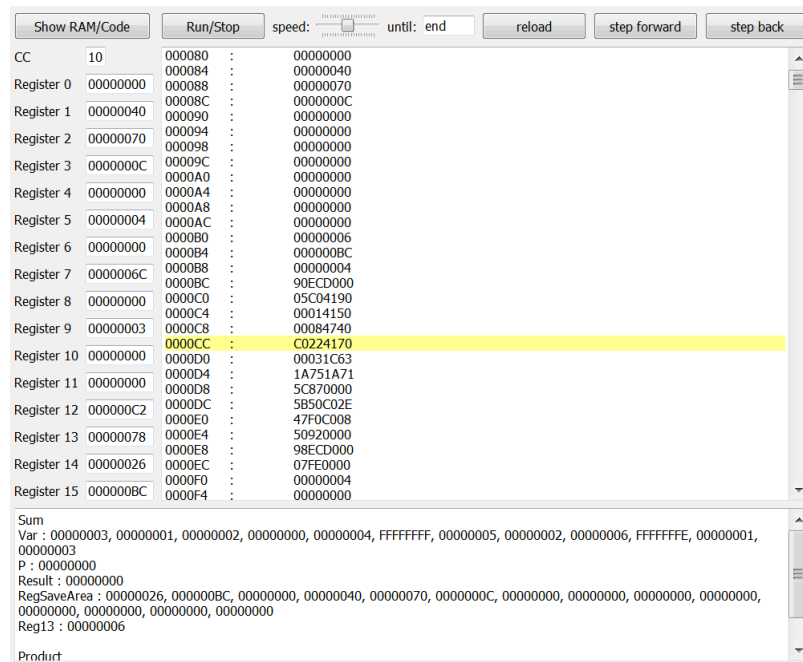
- وضعیت رجیسترها و متغیرها قابل مشاهده است و با زدن دکمه‌ی show RAM/Code می‌توانید

وضعیت حافظه‌ی اصلی را مشاهده کنید؛ تعداد محدودی بایت از اول حافظه نمایش داده می‌شود و آدرسی که در حال

اجراست زردرنگ می‌شود (شکل ۲.۲).



شکل ۲.۱. محیط اجرای برنامه.



شکل ۲.۲. نمایش حافظه‌ی اصلی.

- در پنجره‌ی اجرای کد، می‌توانید سرعت اجرا و آدرس آخرین خطی را که می‌خواهید اجرا شود مشخص کرده و

با زدن دکمه‌ی run/stop اجرای کد را شروع/متوقف کنید؛ هنگام اجرا، خط درحال اجرا زردرنگ می‌شود.

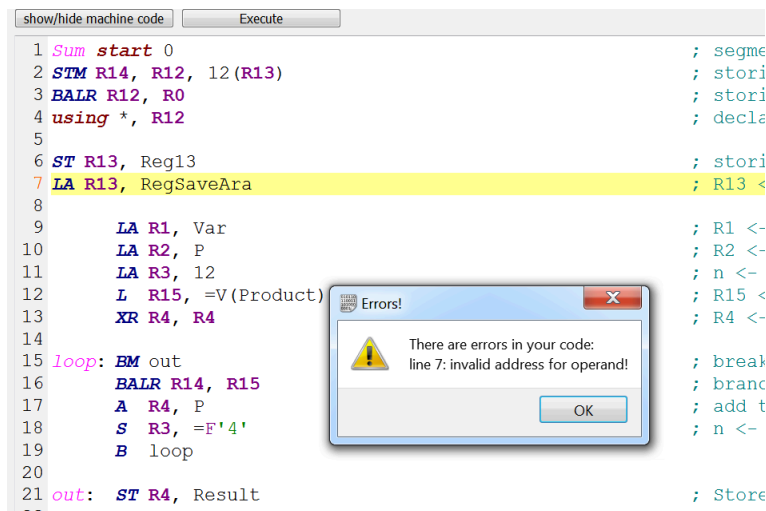
* در اینجا end به این معنی است که تا هرجا که کد کنترل را به سیستم عامل برگرداند اجرا کن.

- step forward، step back و reload به ترتیب وضعیت را به یک مرحله‌ی قبل، یک مرحله‌ی

بعد، و حالت اولیه می‌برند.

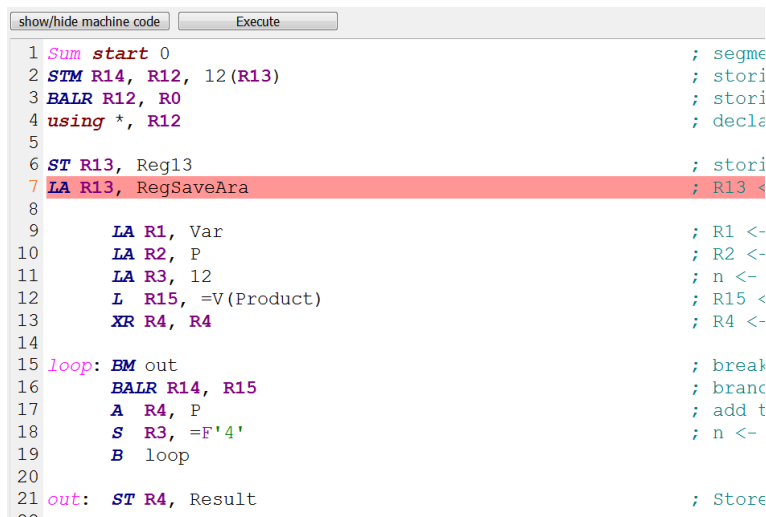
- اگر خطایی درکد وجود داشته باشد، هنگام تلاش برای نمایش کد ماشین و یا بازکردن پنجره‌ی اجرا، پیغام

خطایی نمایش داده شده (شکل ۳.۱) و پس از بستن پنجره‌ی خطا، خط مشکل‌دار قرمز خواهد شد (شکل ۳.۲).



```
show/hide machine code Execute
1 Sum start 0 ; segme
2 STM R14, R12, 12(R13) ; stori
3 BALR R12, R0 ; stori
4 using *, R12 ; decla
5
6 ST R13, Reg13 ; stori
7 LA R13, RegSaveAra ; R13 <
8
9 LA R1, Var ; R1 <-
10 LA R2, P ; R2 <-
11 LA R3, 12 ; n <-
12 L R15, =V(Product) ; R15 <
13 XR R4, R4 ; R4 <-
14
15 loop: EM out ; break
16 BALR R14, R15 ; branch
17 A R4, P ; add t
18 S R3, =F'4' ; n <-
19 B loop
20
21 out: ST R4, Result ; Store
```

شکل ۳.۱. پنجره‌ی خطا.



```
show/hide machine code Execute
1 Sum start 0 ; segme
2 STM R14, R12, 12(R13) ; stori
3 BALR R12, R0 ; stori
4 using *, R12 ; decla
5
6 ST R13, Reg13 ; stori
7 LA R13, RegSaveAra ; R13 <
8
9 LA R1, Var ; R1 <-
10 LA R2, P ; R2 <-
11 LA R3, 12 ; n <-
12 L R15, =V(Product) ; R15 <
13 XR R4, R4 ; R4 <-
14
15 loop: EM out ; break
16 BALR R14, R15 ; branch
17 A R4, P ; add t
18 S R3, =F'4' ; n <-
19 B loop
20
21 out: ST R4, Result ; Store
```

شکل ۳.۲. خط مشکل‌دار.

- کدزدن در این شبیه‌ساز کمی با ماشین اصلی فرق می‌کند، و سعی شده است راحت‌تر و خواناتر باشد.
- در فایل example.txt یک نمونه کد را مشاهده می‌کنید.
- هر سگمنت با `startOffset` `start` `segLabel` شروع و با `end` تمام می‌شود.
 - رجیستر `base` (معمولاً رجیستر ۱۲) را با `baseReg`، `*`، `using` مشخص کنید.
 - * برای آدرس‌دهی صحیح، قبل از تعیین رجیستر `base`، با `R0`، `baseReg`، `BALR` مقدار `PC` را در آن ذخیره کنید (مقدار `R0` همیشه برابر صفر است و در این دستور تاثیری ندارد).
 - می‌توانید قبل از `end`، متغیرهای برنامه را تعریف کنید.
 - می‌توانید به یک خط کد `label` دهید و اسم `label` را با : از کد جدا کنید.
 - * اسم متغیرها و `label`ها می‌تواند شامل حروف و اعداد باشد اما نباید با عدد شروع شود.
 - برای کامنت‌گذاری از ; استفاده کنید.
 - رجیسترها بصورت `RX` نمایش داده می‌شوند تا کد خواناتر باشد.
 - اگر دستوری برای بازگشت کنترل به `OS` وجود نداشته باشد، برنامه ادامه‌ی حافظه را به عنوان کد می‌خواند؛ در نتیجه، بهتر است در انتهای دستورهای اجرایی و قبل از معرفی متغیرها از `R14 BR` استفاده کنید.
 - * معمولاً از رجیستر ۱۴ برای ذخیره‌ی آدرس بازگشت استفاده می‌شود؛ اگر مقدار آن را تغییر ندهید، دستور `BR` کنترل را به سیستم عامل برگردانده و برنامه خاتمه می‌یابد. همچنین می‌توانید از `RX BR` که مقدار رجیستر `X` برابر مقدار اولیه‌ی رجیستر ۱۴ است استفاده کنید.
 - برای اینکه مقادیر رجیسترهای سگمنت والد را در سگمنت فرزند از دست ندهید، می‌توانید در ابتدای سگمنت فرزند با دستور `STM` مقادیر رجیسترها را ذخیره کرده و در انتها با دستور `LM` آن‌ها را بازخوانی کنید.
 - * در ابتدای کار مقدار `R13` برابر آدرس انتهای بخشی از حافظه است که برای ذخیره‌ی رجیسترها معین شده است (stack pointer). در این شبیه‌ساز `OS`ی وجود ندارد که مقادیر رجیسترها را نیاز داشته باشد، پس نیازی به ذخیره‌ی مقادیر نیست.

اما معمولاً، برای اینکه مقادیر رجیسترها بصورت استکمانند (R14, R15, R0, R1, ..., R12) در این حافظه ذخیره شود، در ابتدای اولین سگمنت کاربر با (R13) 12, R12, R14, STM و در انتهای آن با (R13) 12, R12, R14, LM به ترتیب مقادیر را ذخیره و بازخوانی می‌کند.

- این شبیه‌ساز (تقریباً) از تمامی دستورهای که نحوه‌ی کار با آن‌ها در درس ساختار و زبان کامپیوتر تدریس می‌شود پشتیبانی می‌کند.

پشتیبانی

برای گزارش مشکلات یا مطرح نمودن سوال درمورد نحوه‌ی عملکرد شبیه‌ساز با assembly.ta@gmail.com تماس بگیرید.