

# تمرین فصل دوم - دستورات: زبان کامپیوتر

زهره سورانی  
مهدی حق‌وردی

## چکیده

سوالات فصل اول کتاب، که آموزش ISA MIPS است، برای شما تالیف شده‌اند. پاسخ هر سوال را در قسمت مربوط آنها در کوئرا به صورت PDF به صورت تایپ شده، یا دست‌نویس خوش خط و خوانا آپلود کنید. هر سوال دارای یک پاورقی‌ست که طراح آن سوال را مشخص می‌کند، برای پرسیدن سوالات خود به طراح هر سوال مراجعه کنید. پس از پایان یافتن زمان ارسال تمرین، پاسخ‌های این تمرین در آدرس زیر قرار خواهد گرفت.

<https://github.com/mahdihaghverdi/arch-questions-answers/blob/main/instructions-language-of-the-computer>

## فهرست مطالب

۲	۱	C به Assembly
۲	۲	عملیات‌های منطقی
۳	۳	عملیات‌های شرطی (Branches)
۳	۱.۳	مقدار نهایی چیست؟
۳	۲.۳	کد اسمبلی را بنویسید.
۳	۴	توابع
۳	۱.۴	تابع positive
۴	۲.۴	In-line کردن تابع
۴	۳.۴	چندین بار صدا زده شدن تابع
۴	۵	توابع بازگشتی
۴	۱.۵	تابع فاکتوریل
۵	۲.۵	تابع فیبوناچی

## ۱ C به Assembly<sup>۱</sup>

تابع زیر را به زبان اسمبلی بنویسید.

```
1 void copy(int a[], int b[], int n){
2     int i;
3     for(i=0; i!=n; i++) {
4         a[i]=b[i];
5     }
6 }
```

## ۲ عملیات‌های منطقی<sup>۲</sup>

سوالات زیر را بر اساس جدول زیر که محتویات رجیسترهای \$t0 و \$t1 را داراست، بنویسید.

\$t0 = 0xAAAAAAAA	\$t1 = 0x12345678
-------------------	-------------------

۱. مقدار \$t2 بعد از اجرای کد زیر چقدر است؟ پاسخ را به صورت hexadecimal بنویسید.

```
1 sll $t2, $t0, 44
2 or  $t2, $t2, $t1
```

۲. مقدار \$t2 بعد از اجرای کد زیر چقدر است؟ پاسخ را به صورت hexadecimal بنویسید.

```
1 sll $t2, $t0, 4
2 andi $t2, $t2, -1
```

۳. مقدار \$t2 بعد از اجرای کد زیر چقدر است؟ پاسخ را به صورت hexadecimal بنویسید.

```
1 srl $t2, $t0, 3
2 andi $t2, $t2, 0xFFEF
```

---

<sup>۱</sup> مهدی حق‌وردی  
<sup>۲</sup> زهره سورانی

## ۳ عملیات‌های شرطی (Branches)

### ۱.۳ مقدار نهایی چیست؟<sup>۳</sup>

کدهایی زیر را در نظر بگیرید

```
1. 

---

1 LOOP:  addi  $s2, $s2, 2  
2         subi  $t1, $t1, 1  
3         bne   $t1, $0, LOOP  
4 DONE:
```

```
2. 

---

1 LOOP:  slt   $t2, $0, $t1  
2         beq   $t2, $0, DONE  
3         subi  $t1, $t1, 1  
4         addi  $s2, $s2, 2  
5         j     LOOP  
6 DONE:
```

فرض کنید که مقدار اولیه‌ی رجیستر \$t1 برابر است با 10 و مقدار اولیه رجیستر \$s2 برابر با صفر، مقدار نهایی \$s2 چند است؟

### ۲.۳ کد اسمبلی را بنویسید.<sup>۴</sup>

کد اسمبلی کد زیر را بنویسید.

```


---

1 for(i=0; i<a; i++)  
2     for(j=0; j<b; j++)  
3         D[4*j] = i + j;  


---


```

## ۴ توابع

### ۱.۴ تابع positive<sup>۵</sup>

کد اسمبلی تابع زیر را بنویسید.

<sup>۳</sup>زهره سورانی  
<sup>۴</sup>مهدی حق‌وردی  
<sup>۵</sup>زهره سورانی

---

```

1 int positive(int a, int b) {
2     if (addit(a, b) > 0)
3         return 1;
4     else
5         return 0;
6 }
7 int addit(int a, int b) {return a+b;}

```

---

## ۲.۴ In-line کردن تابع<sup>۶</sup>

کامپایلرهای زبان‌های برنامه‌نویسی می‌توانند توابعی را in-line کنند. این به این معناست که کد تابع را در دل جایی که نوشته شده‌اند، تزریق میکنند و از یک function call جلوگیری می‌کنند. حال شما کد تابع addit را in-line کنید.

## ۳.۴ چندین بار صدا زده شدن تابع<sup>۷</sup>

کد زیر را به اسمبلی بنویسید. (نیازی به دانستن بدنه‌ی تابع func نیست، صرفاً آن را به صورت یک label تعریف کنید)

---

```

1 int f(int a, int b, int c, int d){
2     return func(func(a,b), c+d);
3 }

```

---

## ۵ توابع بازگشتی

### ۱.۵ تابع فاکتوریل<sup>۸</sup>

کد اسمبلی زیر، یک کد دارای اشتباه از پیاده‌سازی فاکتوریل است، آنرا اصلاح کنید. مقدار اولیه پاس داده شده به تابع، در رجیستر \$a0 ذخیره شده و نتیجه در رجیستر \$v0 ذخیره می‌شود.

---

```

1 FACT:  sw    $ra, 4($sp)
2         sw    $a0, 0($sp)
3         addi  $sp, $sp, -8
4         slti  $t0, $a0, 1

```

---



---

<sup>۶</sup>زهره سورانی  
<sup>۷</sup>مهدی حق‌وردی  
<sup>۸</sup>زهره سورانی

```

5      beq    $t0, $0, L1
6      addi   $v0, $0, 1
7      addi   $sp, $sp, 8
8      jr     $ra
9
10 L1:  addi   $a0, $a0, -1
11      jal    FACT
12      addi   $sp, $sp, 8
13      lw     $a0, 0($sp)
14      lw     $ra, 4($sp)
15      mul    $v0, $a0, $v0
16      jr     $ra

```

---

## ۲.۵ تابع فیبوناچی<sup>۹</sup>

کد اسمبلی زیر، یک کد دارای اشتباه از پیاده‌سازی دنباله‌ی فیبوناچی است، آنرا اصلاح کنید. مقدار اولیه پاس داده شده به تابع، در رجیستر \$a0 ذخیره شده و نتیجه در رجیستر \$v0 ذخیره می‌شود.

```

1 FIB:  addi   $sp, $sp, -12
2      sw     $ra, 0($sp)
3      sw     $s1, 4($sp)
4      sw     $a0, 8($sp)
5      slti   $t0, $a0, 1
6      beq    $t0, $0, L1
7      addi   $v0, $a0, $0
8      j      EXIT
9
10 L1:  addi   $a0, $a0, -1
11      jal    FIB
12      addi   $s1, $v0, $0
13      addi   $a0, $a0, -1
14      jal    FIB
15      add    $v0, $v0, $s1
16
17 EXIT: lw     $ra, 0($sp)
18      lw     $a0, 8($sp)
19      lw     $s1, 4($sp)
20      addi   $sp, $sp, 12

```

---

<sup>۹</sup> مهدی حق‌وردی

