

به نام خدا



درس ساختار و زبان کامپیوتر
نیم سال اول ۰۴-۰۳
استاد: دکتر اسدی

دانشکده مهندسی کامپیوتر

تمرین سری چهارم

- پرسش‌های خود را در سامانه CW و تالار مربوط به تمرین مطرح نمایید.
- پاسخ سوالات را تایپ نمایید.
- اسکرین‌شات‌ها، عکس‌ها، فایل‌های مربوط به سوال عملی، گزارش تمرینات و PDF قسمت تئوری را در پوشه با نام به فرمت HWNUM_StudentID1_StudentID2 ذخیره نمایید. سپس آن را zip نمایید و در صفحه درس بارگذاری نمایید. به عنوان مثال یک فایل بارگذاری شده قابل قبول باید دارای فرمت HW1_400123456_403123456.zip باشد.
- هر دانشجو می‌تواند حداکثر دو تمرین را با دو روز تأخیر بدون کاهش نمره ارسال نماید.
- تمرینات عملی به صورت گروه‌های دو نفر تحویل داده شود.
- هر دو عضو گروه موظف هستند تمرینات خود را بارگذاری کنند.
- عواقب عدم تطابق بین پاسخ دو عضو گروه برعهده خودشان است.
- تحویل تمرین به صورت انگلیسی مجاز نیست. در صورت تحویل تمرین به صورت انگلیسی (حتی بخشی از تمرین) نمره تمرین موردنظر صفر در نظر گرفته می‌شود.
- در صورت مشاهده تقلب برای بار اول نمره هر دو طرف صفر می‌شود. در صورت تکرار نمره کل تمرینات صفر خواهد شد.
- استفاده از ابزارهایی مانند ChatGPT به منظور ابزار کمک آموزشی مجاز است به شرط آن که به خروجی آن اکتفا نشود.
- توجه شود که پروژه نهایی درس در گروه‌های چهار نفر تحویل گرفته می‌شود.
- سوالات با عنوان اختیاری نمره‌ای ندارند اما جواب دادن به آن‌ها کمک به سزایی در یادگیری درس می‌کند.

تمارین تئوری

۱. کد سی زیر با به زبان اسمبلی MIPS بنویسید، دقت کنید که مجاز به استفاده از دستورهایی شرطی و حلقه نیستید.

```
1 int a[3], b[3], c[3];
2 for (int i = 0; i != 3; i++) {
3     c[i] = 4*a[i] - 8*b[i];
4 }
5 for (int i = 1; i != 3; i++) {
6     c[i] = 2*c[i-1] + c[i];
7 }
```

۲. به سوالات زیر پاسخ دهید:

- (آ) شبه‌دستور^۱ را تعریف کنید و توضیح دهید که چه تفاوتی با دستورات واقعی دارند. دو مثال از این نوع دستورها بنویسید.
- (ب) تحلیل کنید که چرا استفاده از شبه‌دستور می‌تواند برای برنامه‌نویسان MIPS مفید باشد و چه محدودیت‌هایی ممکن است داشته باشد.
- (ج) مقایسه کنید که استفاده از شبه‌دستور چگونه می‌تواند در مقایسه با نوشتن دستورات واقعی بهینه‌سازی کد کمک کند. به این سوال پاسخ دهید که در چه شرایطی استفاده از دستورهایی مجازی ممکن است باعث کاهش عملکرد شود.
- (د) بررسی کنید که آیا استفاده از شبه‌دستور می‌تواند در ایجاد آسیب‌پذیری‌های امنیتی تاثیرگذار باشد یا خیر. آیا ممکن است برخی از دستورهایی مجازی باعث ایجاد رفتارهای غیرقابل پیش‌بینی در کد شوند؟
- (ه) توضیح دهید که دستور move چه عملکردی دارد و به چه صورت به دستورات واقعی ترجمه می‌شود.
- (و) دستور li را توضیح دهید و ذکر کنید که چگونه می‌توان از آن برای بارگذاری مقادیر کمتر و بزرگتر از ۱۶ بیت استفاده کرد.

۳. به سوالات زیر در رابطه با دستورات MIPS پاسخ دهید.

(آ) کد ماشین زیر را به اسمبلی برگردانید و در هر خط، مقدار ثبات مقصد را به صورت دهدهی مشخص کنید.

```
1 0x00842026
2 0x34900001
3 0x001087c0
4 0x00108043
5 0x00102782
```

(ب) کد زیر چه کاری انجام می‌دهد؟ کد ماشین آن را بنویسید.

```
1 and $a0, $s0, $s1
2 xor $a0, $s0, $a0
3 or $a1, $s0, $s1
4 xor $a1, $a1, $s0
5 sub $a0, $a0, $a1
```

(ج) کد زیر چه کاری را انجام می‌دهد؟

```
1 or $a0, $s0, $s1
2 and $a1, $s0, $s1
3 addi $a1, $a1, 1
4 ori $a2, $a2, 1
```

۴. ابتدا دستورات زیر را به اسمبلی تبدیل کنید و بعد از اجرای دستورات زیر مقدار ثبات های تغییر یافته را مشخص کنید.
(اگر به مقدار اولیه ثبات یا حافظه نیاز داشتید صرفاً کافیست آنرا با $mem_i[addr]$ و reg_i مشخص کنید.)

۵. پس از کامپایل کردن کد زیر توسط کامپایلر، GCC کد اسمبلی‌ای که در ادامه قرار گرفته است بدست آمده است. کامپایلر به واسطه سطح بهینه سازی تنظیم شده برای آن، ۳ مورد بهینه سازی بر روی این کد انجام داده است. با توجه به قسمت مشخص شده در کد اسمبلی‌ای که در ادامه قرار داده شده است، بهینه سازی‌های انجام شده را نام برده و به اختصار توضیح دهید هر کدام از این بهینه سازی‌ها باعث جلوگیری از چه سرباری می‌شوند.

کد اسمبلی حاصل از کامپایل به فرم زیر است:

```

1  .data
2  array: .word 13, 2, 43, 24
3  format: .asciiz "Sum: %d\n" # Output format
4
5  .text
6  .globl main
7
8  main:
9      # Prologue
10     addi $sp, $sp, -16          # Create space on stack
11     sw    $ra, 12($sp)         # Save return address
12     sw    $s0, 8($sp)          # Save sum
13     sw    $s1, 4($sp)          # Save array index pointer
14
15     # Initialize variables
16     li    $s0, 0                # sum = 0
17     la    $s1, array            # Load address of the array into $s1
18
19     # Start of the loop
20     lw    $t0, 0($s1)
21     add   $s0, $s0, $t0
22     lw    $t1, 4($s1)
23     add   $s0, $s0, $t1
24     lw    $t2, 8($s1)
25     add   $s0, $s0, $t2
26     lw    $t3, 12($s1)
27     add   $s0, $s0, $t3
28     # End of the loop
29
30     # Print the result
31     li    $v0, 1                # Load syscall for printing integers
32     move  $a0, $s0              # Move the sum into $a0 for printing
33     syscall                     # Print sum
34
35     # Print newline
36     li    $v0, 4                # Load syscall for printing strings
37     la    $a0, format           # Load address of format string
38     syscall                     # Print format string
39
40     # Epilogue
41     lw    $ra, 12($sp)          # Restore return address
42     lw    $s0, 8($sp)          # Restore sum
43     lw    $s1, 4($sp)          # Restore array index pointer
44     addi  $sp, $sp, 16          # Restore stack pointer
45     jr    $ra                  # Return from main

```

تمارین عملی

۱. بدون استفاده از دستورات branch یا هرگونه jump و دستور شرطی، برنامه‌ای بنویسید که تعیین کند آیا مقادیر درون \$t1, \$t2, \$t3 می‌توانند طول اضلاع یک مثلث باشند یا خیر. در صورت جواب مثبت، عدد ۱ را در \$t4 بنویسید و در غیر اینصورت عدد ۰ را بنویسید.

۲. برنامه‌ای به زبان اسمبلی MIPS بنویسید که در آن ابتدا ۳ عدد بین ۰ تا ۳۱ را ورودی بگیرد، سپس مراحل زیر را انجام دهد:

(آ) ابتدا این ۳ عدد را مرتب کرده و خروجی آن‌ها را نمایش دهد.

(ب) مقادیر دودویی^۲ اعداد را معکوس^۳ کرده و با همان ترتیب قبل آن‌ها را نمایش دهد.

توجه داشته باشید که نباید از هیچ دستور پرش، دستور شرطی و همچنین دستورات خارج اسلایدها استفاده کنید.
نمونه‌ی ورودی:

```
1 3 23 7
```

خروجی:

```
1 23 7 3
2 111010 111000 110000
```

۳. برنامه‌ای به زبان اسمبلی MIPS بنویسید که در ورودی، در خط اول یک عدد باینری ۶ بیتی و در خط دوم یک عدد طبیعی (بین ۰ تا ۶۳) دریافت کند. در دو خط اول خروجی، ابتدا عدد باینری را به صورت دهدهی و مبنای ۱۶ و سپس در دو خط بعدی، عدد دهدهی را به صورت باینری و مبنای ۱۶ نمایش دهد.

- ورودی‌ها در بازه‌ی اعداد ۶ بیتی بدون علامت هستند.
- خروجی و ورودی اعداد باینری به صورت ۶ بیتی و خروجی اعداد مبنای ۱۶ باید ۲ رقمی باشد.
- استفاده از دستورات شرطی، حلقه و تقسیم مجاز نیست؛ تنها مجاز به استفاده از دستوراتی که در کلاس آموختید، هستید.
- پیشنهاد می‌شود برای جلوگیری از تکرار دستورات مشابه، از ماکروها (macro) استفاده کنید.

نمونه‌هایی از ورودی و خروجی را می‌توانید در زیر مشاهده کنید:

```
1 input:
2 111111
3 6
4 output:
5 63
6 3f
7 000110
8 06
```

```
1 input:
2 001001
3 0
4 output:
5 9
6 09
7 000000
8 00
```

۴. دستورات sra و srav (شیفت راست حسابی) را بدون دستورات شیفت حسابی و شیفت چپ و ضرب و تقسیم پیاده سازی کنید.

۵. فرض کنید مقادیر a, b, c و d به ترتیب به عنوان ضرایب چندجمله‌ای در حافظه ذخیره شده‌اند. برنامه‌ای در زبان اسمبلی MIPS بنویسید که:

۱. مقدار x را از ورودی کاربر دریافت کند.

۲. مقدار چندجمله‌ای زیر را با استفاده از x محاسبه کند:

$$f(x) = ax^3 + bx^2 + cx + d$$

۳. نتیجه محاسبه شده را به صورت عدد صحیح (integer) در خروجی نمایش دهد.