به نام خدا

دانشکده مهندسی کامپیوتر

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

تمرین سری پنجم

- 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 به‌سزایی در یادگیری درس می‌کند.

تمارین تئوری

۱. کد زیر را تحلیل کنید و خروجی آن را نیز بیان کنید.(فرض کنید که آدرس func2، ۳۲ بیت می‌باشد)

```
.text
main:
    lui     $a0, 1024
    jal     func1
    addi    $a0, $a0, 1
    li      $v0, 1
    syscall
    li      $v0, 10
    syscall

func1:
    xori    $a0, $a0, 1
    j       func2
    addi    $a0, $a0, 1
    jr      $ra

func2:
    xori    $a0, $a0, 1
    addi    $ra, $ra, 4
    lw      $s0, func2 + 36($zero)
    xor     $s0, $s0, $a0
    sw      $s0, func2 + 36($zero)
    jal     func3
    jr      $ra

func3:
    srl     $a0, $a0, 26
    add     $a0, $a0, $a0
    jr      $ra
```

۲. به سوالات زیر پاسخ دهید:

(آ) کد ماشین زیر را به زبان اسمبلی MIPS برگردانید. فرض کنید اولین دستور در آدرس 0x00400000 است. می‌توانید از برچسب دلخواه استفاده کنید.

```
    0x20080000
    0x00044820
    0x200A0001
    0x010A4020
    0x214A0001
    0x112A0001
    0x08100003
```

(ب) کد بالا چه کاری را انجام می‌دهد؟ به ازای تمامی مقادیر ورودی بررسی کنید.

(ج) چه مشکلی در کد بالا وجود دارد؟ به ازای چه مقادیری اجرای کد پایان نمی‌یابد؟

(د) برای رفع مشکل در کد بالا باید چه تغییری در کد ایجاد کرد؟

(ه) درباره انواع دستورات پرش در پردازنده MIPS تحقیق کنید و بگویید آیا دستوری در زبان اسمبلی MIPS وجود دارد که بتوان از طریق آن مشکل موجود در کد را حل کرد؟ اگر جوابتان مثبت است، کد را بازنویسی کنید. نیازی به نوشتن کد ماشین نیست.

۳. شبه‌دستورهای زیر را با رعایت طرز درست استفاده از ثبات‌ها با استفاده از اسمبلی MIPS پیاده‌سازی کنید. کوتاه‌ترین پیاده‌سازی ممکن را ارائه دهید. اجازه استفاده از شبه‌دستورات را ندارید.

برای قسمت آخر (max) کمی درمورد دستورات موجود تحقیق کنید.

setdivis $t0, $t1, $t2 # set t0 to 1 if t1 divisible by t2, 0 otherwise

bitcount $rd, $rs # set rd to the number of 1 bits in rs

circularlshift $rd, $rs, shift # set rd to rs circularly shifted **left** by shift bits

max $rd, $rs, $rt # put the larger of rs or rt into rd

۴. برنامه‌ی اسمبلی زیر را طوری تکمیل کنید که آرایه را به صورت صعودی مرتب کند.

```
.globl main
.data
    array: .word 5, 2, 8, 1, 6 # Example array
    array_size: .word 5

.text
main:
    la $t0, array
    ----- $t1, array_size
    ----- $t2, 0

outer_loop:
    bge $t2, $t1, -----
    li $t3, -----

inner_loop:
    sub -----, $t1, 1
    bge $t3, $t4, end_inner_loop

    sll $t5, $t3, -----
    add $t5, $t5, $t0
    lw $t6, 0($t5)
    lw $t7, -----($t5)

    ble -----, -----, no_swap
    sw -----, -----
    sw -----, -----

no_swap:
    addi $t3, $t3, 1
    j inner_loop

end_inner_loop:
    addi -----, -----, 1
    j outer_loop

end_outer_loop:

    li $v0, 10          # Exit program
    syscall
```

۵. به سوالات زیر پاسخ دهید:

(آ) با فرض اینکه می‌خواهیم ۸ عدد دستور جدید به دستورات MIPS اضافه کنیم بررسی کنید که دستورات R Type، J Type و I Type چه تغییراتی می‌کنند.

(ب) در همان حالت اول فرض کنید که می‌خواهیم ۳۲ ثبات[1] جدید نیز به این پردازنده اضافه کنیم دوباره تغییرات لازم را ذکر کنید.

(ج) حال فرض کنید که می‌خواهیم یک پردازنده‌ی جدید طراحی کنیم که ۳۲ ثبات، ۷۶ کد عملیات[2] و حافظه‌ی ۴ مگابایتی دارد که آدرس دهی به هر بایت آن از طریق ثبات‌ها انجام می‌پذیرد. دستورات این پردازنده ۳ آدرسه و به شکل زیر است:

```
Opcode | Reg1 | Reg2 | Reg3
```

حال مشخص کنید که حداقل طول این دستورات و حداقل اندازه‌ی ثبات‌ها چقدر است.

۲. با توجه به کد زیر به سوالات پاسخ دهید.

```
fun:
bne  a0, $0, norm
sllv t0, a1, a2
bne  t0, $0, norm
li   v0, 0
jr   ra
norm:
sw   ra, 0(sp)
subi sp, 4
li   t0, 32
sub  t0, t0, a2
sllv t0, a0, t0
srlv a0, a0, a2
srlv a1, a1, a2
or   a1, a1, t0
jal  fun
addi v0, v0, 1
addi sp, 4
lw   ra, 0(sp)
jr   ra
```

کد بالا را به زبان c ترجمه کنید نتیجه اجرای fun(۰،۵،۱) و fun(۱،۲،۳) را حساب کنید. اهمیت ریاضیاتی این تابع را مشخص کنید.

[1] register
[2] Opcode

تمارین عملی

۱. در این سوال می‌خواهیم ماشین پشته‌ای[۳] را با استفاده از اسمبلی MIPS پیاده‌سازی کنیم.
در این سوال مجاز به استفاده از پشته MIPS نیستید. همچنین تعداد pushها مشخص نیست و نمی‌توانید از حافظه ایستا[۴] استفاده کنید.
راهنمایی: می‌توانید از sbrk (syscall شماره نهم) برای تخصیص حافظه پویا[۵] استفاده کنید.

این ماشین شامل دستورات زیر می‌باشد:

- psh: زمانی که این دستور می‌آید، در خط بعدی یک عدد صحیح به شما داده می‌شود که این عدد صحیح در پشته push می‌شود.
- pop: این دستور، عدد بالای پشته را pop می‌کند و آن را چاپ می‌کند.
- add: این دستور، دو عدد بالای پشته را pop کرده و حاصل جمع آن‌ها را push می‌کند.
- sub: این دستور، دو عدد بالای پشته را pop کرده و عدد بالاتر را از عدد پایین‌تر کم می‌کند و حاصل تفریق را push می‌کند.
- mul: این دستور، دو عدد بالای پشته را pop کرده و حاصل ضرب آن‌ها را push می‌کند.
- ext: بعد از این دستور، برنامه تمام می‌شود.

نمونه ورودی:

```
psh
4
psh
10
psh
0
psh
-7
sub
add
mul
pop
ext
```

نمونه خروجی:

```
68
```

توضیح تست: در اینجا ابتدا حاصل (۷-) - ۰ حساب شده که برابر با ۷ است سپس ۷ با ۱۰ جمع شده و در نهایت نیز ۱۷ ضرب در ۴ می‌شود و زمانی که pop می‌کنیم، تنها عدد ۶۸ در استک موجود است و این مقدار نیز pop شده و چاپ می‌شود.

۲. کدی بزنید که دو عدد a و b را ورودی بگیرد و a^b را به صورت بازگشتی محاسبه کند. در این محاسبه شما باید از پیچیدگی $O(\log(b))$ استفاده کنید و محاسبه توان به صورت خطی مجاز نیست.

۳. برنامه‌ای به زبان اسمبلی MIPS بنویسید که در آرایه‌ای به طول n، تعداد جفت اعدادی که نسبت به هم اول هستند را محاسبه کند.

---

[۳] Stack Machine
[۴] static
[۵] dynamic

در ورودی، ابتدا عدد n که طول آرایه است داده می‌شود. سپس n عدد که اعضای آرایه هستند ورودی داده می‌شوند. خروجی برنامه تعداد جفت اعدادی که نسبت به هم اول هستند را نشان می‌دهد.

راهنمایی: برای تشخیص اینکه دو عدد نسبت به هم اول هستند یا خیر، می‌توانید از الگوریتم اقلیدسی استفاده کنید.

۴. برنامه‌ای به زبان اسمبلی MIPS بنویسید که ابتدا عدد n را از ورودی دریافت کند. سپس n عدد صحیح را از ورودی روی خطوط مجزا بخواند، و با استفاده از Bubble Sort آنها را از بزرگ به کوچک مرتب کند. سپس آنها را به این ترتیب و در خطوط مجزا در خروجی چاپ کند.

تضمین می‌شود که $n \leq 200$.

۵. برنامه‌ای به زبان اسمبلی MIPS بنویسید که یک ماتریس $N \times M$ را در یک ماتریس $M \times P$ ضرب کند و حاصل را در حافظه ذخیره کند. (می‌توانید فرض کنید اعداد N، M، P و ماتریس‌ها در حافظه ذخیره شده‌اند.)

به عنوان مثال:

```
 .data
N: .word  3
M: .word  2
P: .word  4
A: .word  1, 2, 3, 4, 5, 6          # { {1, 2}, {3, 4}, {5, 6} }
B: .word  8, 7, 6, 5, 4, 3, 2, 1    # { {8, 7, 6, 5}, {4, 3, 2, 1} }
```

۶. یک عدد آرمسترانگ[۶] عددی است که برابر با مجموع ارقام خود باشد، به طوری که هر رقم به توان تعداد ارقام آن عدد رسیده باشد. برنامه‌ای به زبان اسمبلی MIPS پیاده‌سازی کنید که یک عدد هشت بیتی به عنوان ورودی دریافت کرده و در صورتی که عدد آرمسترانگ بود، YES و در غیر این صورت NO چاپ کند.

۷. برنامه‌ای بنویسید که حاصل ضرب دو چندجمله‌ای را حساب و چاپ کند. در ورودی برای هر چند جمله‌ای ابتدا درجه‌ی آن داده می‌شود و سپس ضرایب آن به‌ترتیب از ضریب پرارزش تا کم‌ارزش داده می‌شوند. خروجی شما باید یک رشته با فرمت مناسب باشد:

- جملات با ضریب ۰ نشان داده نشوند.
- ضریب ۱ نشان داده نشود.
- اگر جمله‌ی سمت چپ مثبت بود، علامت آن نشان داده نشود.

مثال:

```
Input:
3
1
1
0
5
2
-2
0
2

This input represents (x^3+x^2+5)(-2x^2+2). Output:

-2x^5-2x^4+2x^3-8x^2+10
```

حداکثر درجه‌ی هر ورودی، ۱۰۰ است. نمونه‌ی ورودی‌ای که کنار برنامه‌ی خود قرار می‌دهید باید رعایت تمام موارد بالا در کدتان را نشان دهد.

[۶] Number Armstrong