به نام خدا

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

دانشکده مهندسی کامپیوتر

تمرین سری هفتم

- 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 عملی 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. به سوالات زیر بر اساس x86 پاسخ دهید.

(آ) در کد زیر در هر دستور مشخص کنید که چه مقداری (به صورت مبنای ۱۶) در ax ذخیره می‌شود و علت آن چیست.

```
segment .data
data: dw 4660, -2
segment .text
mov ax, [data]
mov ax, [data+1]
mov ax, [data+2]
```

(ب) در کد زیر در هر خط، مقدار al یا ah را مشخص کنید.

```
mov al, '8'
add al, '7'
daa
mov ah, '1'
add ah, '9'
aaa
```

(ج) هر یک از کدهای زیر را به یک دستور x86 تبدیل کنید.

```
bsr dx, ax
cmp dx, 15
jne L1
mov dx, -1
jmp L2
L1: xor dx, dx
L2:
```

```
cmp eax, ebx
je  L1
mov eax, ebx
jmp L2
L1:
mov ebx, edx
L2:
```

```
push    ax
xor     rsi, rsi
and     al, 0x80
shr     al, 7
cmp     al, 1
pop     ax
jne      L2
or      rsi, 0xFFFFFFFFFFFFFF00
or      rsi, rax
jmp     done
L2:
and     rax, 0x00000000000000FF
or      rsi, rax
done:
```

۲. تحقیق کنید اگر یک برنامه به زبان C داشته باشیم که از یک کتابخانه از نوع dynamically-linked استفاده کند و توابع موجود در این نوع کتابخانه فراخوانی شوند، مراحل compiling ، assembling ، linking و loading چگونه پیش خواهند رفت؟ فرآیند address resolution برای تابعی که در یک DLL است به چه صورت خواهد بود؟ در این ارتباط بطور کامل چگونگی ایجاد symbol table ها و relocation table ها را تشریح کنید.
پاسخ شما باید توضیحات کاملی از تعریف و دلیل استفاده از مفاهیمی مثل PLT و GOT را شامل شود.

۳. به سوالات زیر پاسخ دهید:

(آ) برای کاربردهای واقعی زیر، مشخص کنید کدام روش آدرس‌دهی بهینه‌تر است و چرا:

- دسترسی به آرایه‌ای از داده‌های مرتب‌شده.
- پیاده‌سازی یک ماشین‌حساب ساده با جمع و تفریق.
- خواندن داده‌ها از یک حافظه چندلایه.

(ب) روش‌های آدرس‌دهی مقیاسی[۱] و خودکار[۲] در معماری‌های پیچیده‌تر مانند x۸۶ وجود دارند. تحقیقی انجام دهید و توضیح دهید که این روش‌ها چگونه می‌توانند در بهبود کارایی پردازنده در مقایسه با معماری ساده‌ای مثل MIPS مؤثر باشند.

(ج) کامپایلرهای مختلف (مانند کامپایلرهای بهینه‌کننده[۳]، کامپایلرهای چندمرحله‌ای[۴]، و کامپایلرهای هم‌زمان با اجرا[۵]) را در زمینه‌های زیر مقایسه کنید:

- پیچیدگی طراحی و پیاده‌سازی
- کارایی نهایی برنامه تولید شده
- مناسب بودن برای سیستم‌های توکار[۶]

(د) مفسرها و کامپایلرها را در موارد زیر مقایسه کنید:

- سرعت اجرای برنامه
- اندازه‌ی کد
- نیاز به منابع سخت‌افزاری

۴. برای هرکدام از موارد زیر، ضمن مشخص کردن مود آدرس‌دهی و همچنین عملکرد دستور، کدی به زبان اسمبلی MIPS ارائه دهید که عملکرد مشابهی در زبان میپس داشته باشد. (امکان استفاده از ثبات اضافی وجود دارد)

```
MOV R1, (R2)
```

```
LOAD R1, (0x2000)
```

```
ADD R1, @(R2)
```

```
SUB R1, 8(R2)[R3]
```

```
PUSH (R1)+
```

```
STORE R1, (R1+R2)
```

---

[۱] Scaled Addressing
[۲] Auto-Increment/Decrement Addressing
[۳] Optimizing Compiler
[۴] Multi-pass Compiler
[۵] Just In Time
[۶] Embedded Systems

۵. آدرس‌دهی‌های Indexed، Auto-Decrement و Scaled را در نظر بگیرید:

الف) برای هر کدام از این حالت‌های آدرس‌دهی[۷]، یک برنامه مثال بزنید که استفاده از آن‌ها نسبت به حالت‌های آدرس‌دهی در MIPS مؤثرتر و کارآمدتر باشد. توضیح دهید چرا این حالت‌ها در این سناریوها مناسب‌تر هستند و چه مزایایی نسبت به MIPS ارائه می‌دهند.

ب) آیا محدودیت‌های حالت‌های آدرس‌دهی در MIPS به دلیل کاهش پیچیدگی سخت‌افزار توجیه‌پذیر است؟ با توجه به کاربردهای خاص، توضیح دهید که افزودن یکی از این حالت‌های آدرس‌دهی (مثلاً Indexed) چگونه می‌تواند عملکرد معماری MIPS را بهبود دهد و چه هزینه‌های سخت‌افزاری ممکن است ایجاد کند.

۶. فرض کنید پردازنده‌ای از حالت‌های آدرس‌دهی غیرمستقیم ثباتی[۸]، مستقیم[۹]، غیرمستقیم حافظه[۱۰]، مقیاس‌بندی شده[۱۱]، افزایش خودکار[۱۲]، کاهش خودکار[۱۳] و شاخص‌دار[۱۴] پشتیبانی می‌کند. برنامه‌ی زیر مجموع تمامی عناصر یک آرایه داده را محاسبه کرده و نتیجه را در حافظه ذخیره می‌کند. شبه‌کد آن به صورت زیر است:

```
Memory[SUM] = 0
For i = 0 to N - 1:
    Memory[SUM] += Data[i]
End For
```

کد اسمبلی آن به صورت زیر است:

```
        .data
SUM:    .word 0
DATA:   .word 5, 8, 3, 12
N:      .word 4

        .text
        .global _start

_start:
        LDR  R1, =SUM           ; Load address of SUM
        MOV  R0, #0             ; Initialize R0 (temporary sum) to 0
        LDR  R2, =DATA
        LDR  R3, =N
        LDR  R3, [R3]

LOOP:
        LDR  R4, [R2], #4
        ADD  R0, R0, R4
        SUBS R3, R3, #1
        BNE  LOOP

        STR  R0, [R1]
        B    END

END:
        NOP                     ; Placeholder for program termination
```

[۷] Addressing Modes
[۸] Indirect Register
[۹] Direct
[۱۰] Indirect Memory
[۱۱] Scaled
[۱۲] Auto-increment
[۱۳] Auto-decrement
[۱۴] Indexed

با توجه به توضیحات فوق، به سوالات زیر پاسخ دهید.

(آ) توضیح دهید که هر حالت آدرس‌دهی چگونه در این برنامه استفاده شده است.

(ب) حلقه را طوری تغییر دهید که از حالت‌های آدرس‌دهی غیر مستقیم حافظه‌ای، ایندکسی و مقیاس‌شده به جای افزایش خودکار استفاده کند.

(ج) کد معادل برای سیستمی که از افزایش خودکار پشتیبانی نمی‌کند، بنویسید.

تمارین عملی

۱. برنامه‌ای به زبان اسمبلی 8086 بنویسید که ابتدا ۴ عدد را به صورت دهدهی ورودی می‌گیرد و سپس دستوراتی به فرمت‌های زیر، به عنوان ورودی به کد داده شده و کد باید مقدار مناسب را خروجی دهد. ورودی‌ها در بازه اعداد علامت‌دار ۱۶ بیتی می‌باشند.

- **cmp num1 num2**: عدد بزرگتر را در خروجی چاپ می‌کند. (در صورت برابری نیز عدد را چاپ کند)
- **swap num1 num2**: مقادیر num1 و num2 را با یکدیگر عوض می‌کند.
- **mul num1 num2**: حاصل ضرب دو عدد را در خروجی نمایش می‌دهد.
- **div num1 num2**: num1 را بر num2 تقسیم کرده و حاصل تقسیم و باقی‌مانده را به صورت صحیح در خروجی نمایش می‌دهد. (در صورتی که مقسوم‌علیه برابر با صفر بود، ارور مناسبی چاپ کنید)
- **msb num**: اندیس اولین بیت از سمت چپ عدد که برابر با یک می‌باشد را خروجی می‌دهد. (zero-index، تضمین می‌شود که ورودی برابر با صفر نیست)
- **lsb num**: اندیس اولین بیت از سمت راست که برابر با یک می‌باشد را خروجی می‌دهد. (شرابط مانند msb می‌باشد)
- **overflow num1 num2**: اگر حاصل جمع این دو عدد سرریز داشت، YES و در غیر این صورت NO چاپ شود.
- **exit**: بعد از این دستور برنامه خاتمه می‌یابد.

نمونه:

```
input:
12
8
5
-6
mul 0 1
swap 0 1
mul 0 3
div 3 1
lsb 2
msb 3
exit
output:
96
-48
0
-6
0
15
```

۲. برنامه‌ای به زبان اسمبلی 8086 بنویسید که یک رشته ورودی بگیرد و بزرگترین زیررشته بدون کاراکترهای تکراری را خروجی دهد. برنامه نباید case sensitive باشد. ورودی نمونه:

```
abcaBcbadAb
```

خروجی نمونه:

```
cbad
```

۳. برنامه‌ای به زبان اسمبلی 8086 بنویسید که یک رشته را در ورودی بگیرد و در خروجی تمام جایگشت‌های ممکن برای حروف آن را به ترتیب حروف الفبا چاپ کند. قرار نیست از ترتیب استاندارد حروف الفبا استفاده کنید، بلکه فرض کنید حروف رشته‌ی ورودی به ترتیب اولویتشان در الفبا آمده‌اند.

مثلا اگر در ورودی ba را داریم، خروجی زیر انتظار می‌رود:

```
ba
ab
```

۴. یک برنامه اسمبلی 8086 بنویسید تا دو رشته‌ی پایان‌یافته با null به نام‌های string1 و string2 را مقایسه کند و بررسی کند که آیا این دو رشته یکسان هستند یا نه. برنامه باید رشته‌ها را بایت به بایت مقایسه کند.

۵. برنامه‌ای به زبان ۸۰۸۶ بنویسید که تعداد 0 و 1های رشته ورودی را در خروجی چاپ کند. ورودی نمونه:

```
"00011100"
```

خروجی نمونه:

```
5
3
```