1. **زبان اسمبلی چیست؟**
زبان اسمبلی یک زبان برنامه‌نویسی سطح پایین است که مستقیماً با سخت‌افزار در تعامل است. این زبان به صورت دستورات متنی ساده طراحی شده که هر دستور معمولاً متناظر با یک دستور زبان ماشین خاص است. اسمبلی به برنامه‌نویسان اجازه می‌دهد کدهای سریع‌تر و بهینه‌تر برای سخت‌افزار خاص بنویسند.
2. **مزایا و معایب زبان اسمبلی را بنویسید:**
    - **مزایا:**
        - سرعت بالای اجرا.
        - دسترسی مستقیم به سخت‌افزار.
        - استفاده بهینه از منابع سیستم.
        - قابلیت نوشتن برنامه‌های کوچک و کارآمد.
    - **معایب:**
        - پیچیدگی و دشواری درک و یادگیری.
        - وابستگی به معماری سخت‌افزار.
        - زمان‌بر بودن برنامه‌نویسی.
        - سختی نگهداری و اشکال‌زدایی.
3. **چهار معماری کامپیوتر رایج و زبان‌های اسمبلی مناسب آن‌ها را نام ببرید؟**
    - **x86:** زبان اسمبلی x86.
    - **ARM:** زبان اسمبلی ARM.
    - **MIPS:** زبان اسمبلی MIPS.
    - **PowerPC:** زبان اسمبلی PowerPC.
4. **عموماً چه برنامه‌هایی را به زبان اسمبلی می‌نویسند؟**
    - برنامه‌های سطح پایین مانند درایورهای سخت‌افزار.
    - سیستم‌های عامل و بخش‌هایی از کرنل.
    - برنامه‌هایی که نیاز به سرعت بالا دارند، مثل الگوریتم‌های پردازش سیگنال یا بازی‌های ویدیویی.
5. **از زبان‌های اسمبلی غالباً در چه حوزه‌ای استفاده می‌شود؟**
    - برنامه‌نویسی سیستم‌ها.
    - توسعه نرم‌افزارهای نهفته.(Embedded Systems)
    - دیباگینگ و تحلیل نرم‌افزار.
    - توسعه نرم‌افزارهای وابسته به سخت‌افزار خاص.

6.تفاوت Compiler و Interpreter چیست؟

- **Compiler) کامپایلر :(**کد منبع را به صورت کامل به زبان ماشین ترجمه می‌کند و فایل اجرایی تولید می‌کند. سرعت اجرای برنامه بالاست، زیرا ترجمه قبلاً انجام شده است.
- **Interpreter) مفسر :(**کد منبع را خط به خط اجرا می‌کند و مستقیماً ترجمه را انجام می‌دهد. سرعت اجرا پایین‌تر است زیرا ترجمه هم‌زمان با اجرا انجام می‌شود.

7.Linker چیست و چه کاری انجام می‌دهد؟

- Linkerبرنامه‌ای است که فایل‌های شیء (Object Files) را با هم ترکیب کرده و یک فایل اجرایی تولید می‌کند. این ابزار کدهای کتابخانه‌ها را نیز به برنامه متصل می‌کند.

---

8 .Object File چیست و شامل چه محتوایی می‌باشد؟

- فایل شیء (Object File) فایل خروجی تولیدشده توسط کامپایلر است. این فایل شامل کد ماشین، داده‌ها و اطلاعات لازم برای اتصال (Linking) است.

---

9. تبدیل اعداد از مبنای 10 به مبناهای 2، 8 و 16 با مثال:

- عدد 25:
  - مبنای 2: 11001
  - مبنای 8: 31
  - مبنای 16: 19

---

10. تبدیل اعداد از مبناهای 2، 8 و 16 به 10 با مثال:

- عدد 11001 (مبنای 2): 25
- عدد 31 (مبنای 8): 25
- عدد 19 (مبنای 16): 25

---

11. تفاوت سیستم کدگذاری ASCII و Unicode چیست؟

- **ASCII:**از 7 یا 8 بیت استفاده می‌کند و 128 یا 256 کاراکتر را پوشش می‌دهد.
- **Unicode:**از 16، 32 یا بیشتر بیت استفاده می‌کند و هزاران کاراکتر از زبان‌های مختلف دنیا را شامل می‌شود

---

12. تفاوت کامپیوترهای RISC و CISC چیست؟

- **RISC (Reduced Instruction Set Computer):** دستورات ساده و سریع؛ مناسب برای پردازش‌های سبک.
- **CISC (Complex Instruction Set Computer):** دستورات پیچیده‌تر؛ نیاز به سخت‌افزار قوی‌تر.

13.حافظه Register چیست و چه کاربردی دارد؟

- رجیستر حافظه‌ای کوچک و سریع در پردازنده است که داده‌های موقت و متغیرها را ذخیره می‌کند.

---

14. Flag Registerچیست؟ عملکرد هر یک از بیت‌های Flag را بیان کنید؟

Flag Registerیا ثبات پرچم در پردازنده‌ها بخشی از ثبات‌ها است که وضعیت‌های خاصی از عملیات حسابی و منطقی را ذخیره می‌کند. این ثبات به‌طور معمول شامل چند بیت است که هر کدام به یک حالت یا وضعیت مشخص اشاره دارند. این بیت‌ها توسط پردازنده تنظیم یا پاک می‌شوند تا نتایج عملیات‌ها را نشان دهند و در تصمیم‌گیری‌های برنامه مانند پرش شرطی استفاده شوند.

ساختار و عملکرد بیت‌های اصلی:Flag Register

Flag Registerدر پردازنده‌های مختلف مانند x86 یا ARM ساختار متفاوتی دارد، اما در معماریx86 ، رایج‌ترین بیت‌های آن عبارتند از:

1. **Carry Flag (CF)**
   - **وظیفه :این بیت نشان‌دهنده سرریز (Carry) در عملیات جمع و یا قرض گرفتن در عملیات تفریق است.**
   - **مقدار:**
     - **اگر سرریز وجود داشته باشد1 :**
     - **در غیر این صورت0 :**
2. **Parity Flag (PF)**
   - **وظیفه :نشان‌دهنده زوج یا فرد بودن تعداد بیت‌های ۱ در نتیجه عملیات است.**
   - **مقدار:**
     - **اگر تعداد بیت‌های ۱ زوج باشد1 :**
     - **اگر فرد باشد0 :**
3. **Auxiliary Carry Flag (AF)**

- وظیفه :نشان‌دهنده سرریز از بیت 3 به بیت 4 در عملیات BCD (Binary-Coded Decimal).
- کاربرد :در عملیات‌های دهدهی.

4. Zero Flag (ZF)
   - وظیفه :مشخص می‌کند که نتیجه عملیات صفر شده است یا خیر.
   - مقدار:
     - اگر نتیجه صفر باشد1 :
     - در غیر این صورت0 :
5. Sign Flag (SF)
   - وظیفه :نشان‌دهنده علامت نتیجه عملیات است.
   - مقدار:
     - اگر نتیجه منفی باشد (بیت بیشترین ارزش ۱ باشد)1 :
     - اگر مثبت باشد0 :
6. Overflow Flag (OF)
   - وظیفه :نشان‌دهنده سرریز در عملیات محاسباتی است، زمانی که نتیجه خارج از بازه قابل نمایش در قالب مکمل ۲ باشد.
   - مقدار:
     - اگر سرریز وجود داشته باشد1 :
     - در غیر این صورت0 :
7. Interrupt Enable Flag (IF)
   - وظیفه :فعال یا غیرفعال کردن وقفه‌ها.
   - مقدار:
     - اگر وقفه‌ها فعال باشند1 :
     - اگر غیرفعال باشند0 :
8. Direction Flag (DF)
   - وظیفه :تعیین جهت پردازش داده‌ها در دستورهای مربوط به رشته.
   - مقدار:
     - اگر مقدار 1باشد، پردازش از انتها به ابتدا (کاهشی).
     - اگر مقدار 0باشد، پردازش از ابتدا به انتها (افزایشی).
9. Trap Flag (TF)

- **وظیفه :فعال کردن حالت Debug یا.Single-step**
- **کاربرد :برای بررسی خط به خط کد.**

15.عملکرد ثبات‌های عمومیEAX ، EBX، ECXو EDX چیست؟

- **EAX:**برای عملیات اصلی و ذخیره نتایج.
- **EBX:**برای آدرس‌دهی.
- **ECX:**به عنوان شمارنده.
- **EDX:**برای عملیات تقسیم یا چندگانه.

---

16. عملکرد عملگرهای محاسباتیadd ، sub، mulو div با مثال:

- **add:** جمع.
  مثال: add eax, 5 → eax = eax + 5
- **sub:** تفریق.
  مثال: sub eax, 5 → eax = eax - 5
- **mul:** ضرب.
  مثال: mul eax, 5 → eax = eax * 5
- **div:** تقسیم.
  مثال: div eax, 5 → eax = eax / 5

---

17. عملکرد دستور cbw با مثال:

- دستور **cbw**علامت یک مقدار 8 بیتی را به مقدار 16 بیتی گسترش می‌دهد.
  مثال: اگر5- = AL ، پس از اجرایcbw ، .5- = AX

---

18. دستور LEA چه کاری انجام می‌دهد؟

- دستور **LEA (Load Effective Address)**آدرس مؤثر یک متغیر یا داده را در یک رجیستر ذخیره می‌کند.

---

19. عملکرد دستور inc و dec با مثال:

- **inc:** افزایش مقدار.
  مثال: inc eax → eax = eax + 1

- **dec:** کاهش مقدار.
  مثال: dec eax → eax = eax - 1

---

20 .دستور XCHG چه کاری انجام می‌دهد؟

- **XCHG:** مقادیر دو رجیستر را با هم تعویض می‌کند.
  مثال: xchg eax, ebx → مقدار eax و ebx جابجا می‌شوند.

---

21 . در یک برنامه ساده عملکرد دستور cmp و یکی از ساختارهای جهش(jump) را بیان کنید؟

دستور :CMP

دستور CMPدو مقدار را با هم مقایسه می‌کند. این مقایسه منجر به تنظیم فلگ‌ها (Flags) در رجیستر وضعیت پردازنده می‌شود، اما هیچ تغییری روی مقادیر اصلی ایجاد نمی‌کند. این فلگ‌ها مشخص می‌کنند که نتیجه مقایسه به چه صورت بوده است (مثلاً یکی بزرگ‌تر است، برابرند، یا یکی کوچک‌تر است).

CMP operand1, operand2

ساختار جهش:(Jump)

دستورات جهش برای تغییر جریان اجرای برنامه استفاده می‌شوند. یکی از دستورهای پرکاربرد **JMP** (Jump)است که اجرای برنامه را به آدرس مشخصی منتقل می‌کند.

توضیح عملکرد CMPو یک جهش (مانند :(JE

1. **CMP**:این دستور دو مقدار را مقایسه می‌کند و فلگ‌های مربوط به وضعیت) مانند Zero Flag را تنظیم می‌کند.
2. **ساختار جهش** :**(Jump)** با توجه به نتیجه تنظیم فلگ‌ها، برنامه می‌تواند به آدرس خاصی بپرد (در اینجا از JEاستفاده می‌شود.(

این برنامه دو عدد را با هم مقایسه می‌کند و اگر برابر باشند، پیام "برابرند" را نمایش می‌دهد.

```
section .data
    msg db "Numbers are equal!", 0Ah  ; پیام مقادیر برابر
    msg_len equ $ - msg               ; طول پیام

section .text
    global _start
```

```
_start:
    mov eax, 5      ; مقدار اول
    mov ebx, 5      ; مقدار دوم

    cmp eax, ebx    ; مقایسه مقدار اول با مقدار دوم
    je equal        ; اگر برابر بودند، به برچسب equal برو

    ; اگر مقادیر برابر نبودند، برنامه در این مثال هیچ کاری نمی‌کند

equal:
    mov edx, msg_len ; طول پیام
    mov ecx, msg     ; آدرس پیام
    mov ebx, 1       ; چاپ روی استاندارد خروجی
    mov eax, 4       ; شماره فراخوان سیستمی برای نوشتن
    int 0x80         ; فراخوانی سیستم برای چاپ

    ; پایان برنامه
    mov eax, 1       ; شماره فراخوان سیستمی برای خروج
    xor ebx, ebx     ; مقدار بازگشتی صفر
    int 0x80         ; فراخوانی سیستم برای خروج
```

تحلیل برنامه:

1. **mov eax, 5** و **mov ebx, 5**: دو مقدار برای مقایسه در رجیسترها بارگذاری می‌شوند.
2. **cmp eax, ebx**:
   - مقدار eax و ebx مقایسه می‌شوند.
   - اگر eax == ebx، فلگ Zero روشن می‌شود.
3. **je equal**:
   - اگر فلگ Zero روشن باشد (مقادیر برابر باشند)، اجرای برنامه به برچسب equal می‌پرد.
4. در برچسب equal پیام "Numbers are equal!" چاپ می‌شود.

22.برنامه‌ای برای چاپ یک Message بر روی Screen:

```
section .data
    msg db "Hello, World!", 0Ah0) پیام برای چاپ ;  Ah = newline(
    msg_len equ $ - msg طول پیام ;
```

```
section .text
    global _start

_start:
; تنظیم پارامترهای فراخوانی سیستم
    mov eax, 4) شماره فراخوان سیستمی برای نوشتن ;       sys_write(
    mov ebx, 1(استاندارد خروجی = 1) شماره فایل دیسکریپتور ;
    mov ecx, msgآدرس پیام ;
    mov edx, msg_lenطول پیام ;
    int 0x80فراخوانی سیستم ;

; پایان برنامه
    mov eax, 1) شماره فراخوان سیستمی برای خروج ;       sys_exit(
    xor ebx, ebxمقدار بازگشتی صفر ;
    int 0x80فراخوانی سیستم ;
```

توضیحات برنامه:

1. **بخش داده‌ها:(section .data)**
   - پیام مورد نظر ("Hello, World!") در بخش داده‌ها تعریف شده است.
   - طول پیام با استفاده از دستور equمحاسبه می‌شود.
2. **بخش کد:(section .text)**
   - در رجیستر **eax**شماره فراخوان سیستم برای نوشتن (sys_write) قرار می‌گیرد.
   - **ebx**شماره فایل دیسکریپتور (1 برای استاندارد خروجی) تنظیم می‌شود.
   - **ecx**آدرس پیام و **edx**طول پیام را نگه می‌دارند.
3. **int 0x80**:
   - این دستور باعث اجرای فراخوانی سیستم می‌شود و پیام روی صفحه چاپ می‌شود.
4. **پایان برنامه**:
   - شماره فراخوان سیستم برای خروج (sys_exit) در **eax**تنظیم می‌شود.
   - برنامه با مقدار بازگشتی صفر خاتمه می‌یاب

23. برنامه ای بنویسید که یک عدد را برحسب متر از صفحه کلید خونده و به سانتیمتر تبدیل نماید؟

```
.MODEL SMALL

.STACK 100h
```

```
.DATA
    message1 DB 'Enter length in meters'$ :
    message2 DB 13, 10, 'Length in centimeters'$ :
    inputNum DB ذخیره ورودی کاربر ;          ?
    result DW(سانتی‌متر) ذخیره نتیجه ;          ?

.CODE
MAIN PROC
تنظیم سگمنت دیتا ;
    MOV AX, @DATA
    MOV DS, AX

نمایش پیام درخواست ورودی ;
    LEA DX, message1
    MOV AH, 9
    INT 21h

دریافت ورودی کاربر ;
    MOV AH, 1
    INT 21h
    SUB AL, '0 تبدیل کاراکتر به عدد ;          '
    MOV BL, AL ذخیره عدد در ;          BL

(100 * عدد) محاسبه سانتی‌متر ;
    MOV AL, BL
    MOV CL, 100
    MUL CL          ; AL * 100 نتیجه در <- AX
    MOV result, AX

نمایش پیام نتیجه ;
    LEA DX, message2
    MOV AH, 9
    INT 21h
```

```
; تبدیل و نمایش عدد
    MOV AX, result
    CALL PRINT_NUM ; نمایش عدد

; پایان برنامه
    MOV AH, 4Ch
    INT 21h

; زیرروال نمایش عدد
PRINT_NUM PROC
    PUSH AX
    XOR CX, CX

ConvertToString:
    XOR DX, DX
    MOV BX, 10
    DIV BX          ; AX / 10 خارج قسمت در AX -> باقی‌مانده در DX
    ADD DL, '0'
    PUSH DX
    INC CX
    TEST AX, AX
    JNZ ConvertToString

DisplayResult:
    POP DX
    MOV AH, 2
    INT 21h
    LOOP DisplayResult

    POP AX
    RET
PRINT_NUM ENDP

MAIN ENDP
```

```
END MAIN
```

24. برنامه ای بنویسید که دو عدد را از ورودی دریافت کرده سپس حاصل جمع آنها را چاپ نماید؟

```
section .data
   prompt1 db "Enter first number: ", 0
   prompt2 db "Enter second number: ", 0
   result db "Sum: ", 0

section .bss
   num1 resb 4
   num2 resb 4

section .text
   global _start
_start:
   ; خواندن عدد اول
   mov eax, 4
   mov ebx, 1
   mov ecx, prompt1
   mov edx, 20
   int 0x80

   mov eax, 3
   mov ebx, 0
   mov ecx, num1
   mov edx, 4
   int 0x80

   ; خواندن عدد دوم
   mov eax, 4
   mov ebx, 1
   mov ecx, prompt2
   mov edx, 20
```

```
int 0x80

mov eax, 3
mov ebx, 0
mov ecx, num2
mov edx, 4
int 0x80

; جمع دو عدد
mov eax, dword [num1]
add eax, dword [num2]

; نمایش نتیجه
mov eax, 4
mov ebx, 1
mov ecx, result
mov edx, 10
int 0x80

; خروج
mov eax, 1
xor ebx, ebx
 int 0x80
```

25.برنامه ای بنویسید که دو عدد را از ورودی دریافت کرده سپس حاصل ضرب آنها را چاپ نماید؟

```
section .data
    prompt1 db "Enter first number: ", 0
    prompt2 db "Enter second number: ", 0
    result db "Product: ", 0

section .bss
```

```
    num1 resb 4
    num2 resb 4

section .text
    global _start
_start:
    ; خواندن عدد اول
    mov eax, 4
    mov ebx, 1
    mov ecx, prompt1
    mov edx, 20
    int 0x80

    mov eax, 3
    mov ebx, 0
    mov ecx, num1
    mov edx, 4
    int 0x80

    ; خواندن عدد دوم
    mov eax, 4
    mov ebx, 1
    mov ecx, prompt2
    mov edx, 20
    int 0x80

    mov eax, 3
    mov ebx, 0
    mov ecx, num2
    mov edx, 4
    int 0x80

    ; ضرب دو عدد
    mov eax, dword [num1]
    imul eax, dword [num2]
```

```
; نمایش نتیجه
mov eax, 4
mov ebx, 1
mov ecx, result
mov edx, 10
int 0x80

; خروج
mov eax, 1
xor ebx, ebx
int 0x80
```