

1. Select an ADD instruction that will:

(a) add BX to AX

English: ADD AX, BX

Bangla: এখানে AX-তে BX-এর মান যোগ করতে চাইলে, ADD AX, BX ব্যবহার করতে হবে। কারণ, এই নির্দেশে  $AX = AX + BX$  হয়।

(b) add 12H to AL

English: ADD AL, 12H

Bangla: AL-এ 12H যোগ করতে চাইলে, ADD AL, 12H ব্যবহার করতে হবে। কারণ, এতে  $AL = AL + 12H$  হয়।

(c) add EDI to EBP

English: ADD EBP, EDI

Bangla: EBP-তে EDI যোগ করতে চাইলে, ADD EBP, EDI ব্যবহার করতে হবে। এতে  $EBP = EBP + EDI$  হয়।

(d) add 20H to EAX

English: ADD EAX, 20H

Bangla: EAX-এ 20H যোগ করতে চাইলে, ADD EAX, 20H ব্যবহার করতে হবে। এতে  $EAX = EAX + 20H$  হয়।

(e) add the data addressed by SI to AL

English: ADD AL, [SI]

Bangla: SI বৈধ মেমরি লোকেশন দেখায়, সেই তেটা AL-এ যোগ করতে চাইলে, ADD AL, [SI] ব্যবহার করতে হবে। এতে  $AL = AL + [SI]$  হয়।

(f) add CX to the data stored at memory location FROG

English: ADD FROG, CX

Bangla: FROG নামের মেমরি লোকেশনে CX যোগ করতে চাইলে, ADD FROG, CX ব্যবহার করতে হবে। এতে  $FROG = FROG + CX$  হয়।

(g) add 234H to RCX

English: ADD RCX, 234H

Bangla: RCX-এ 234H যোগ করতে চাইলে, ADD RCX, 234H ব্যবহার করতে হবে। এতে  $RCX = RCX + 234H$  হয়।

2. What is wrong with the ADD RCX, AX instruction?

English: You cannot use mixed size registers.

Bangla: এখানে RCX (৬৪-বিট) আর AX (১৬-বিট) একসাথে যোগ করা যায় না, কারণ দুইটার সাইজ আলাদা। তাই এই নির্দেশ ভুল।

3. Is it possible to add to CX with the ADD instruction?

English: No instruction is available to add to a segment register.

Bangla: কোনো ADD নির্দেশ দিয়ে সেগমেন্ট রেজিস্টারে (যেমন CX) সরাসরি যোগ করা যায় না। তাই এটা সম্ভব নয়।

4. If AX = 100H and DX = 20FFH, list the sum and the contents of each flag register bit (C, A, S, Z, O) after the ADD AX, DX instruction executes.

English:

After ADD AX, DX:

AX = 3100H, C = 0, A = 1, S = 0, Z = 0, O = 0

Bangla:

AX + DX = 100H + 20FFH = 3100H

- Carry (C) = 0 (কোনো ক্যারি হয়নি)
- Auxiliary Carry (A) = 1 (সোয়াচ ৪ বিটে ক্যারি হয়েছে)
- Sign (S) = 0 (ফলাফল পজিটিভ)
- Zero (Z) = 0 (ফলাফল শূন্য নয়)
- Overflow (O) = 0 (ওভারফ্লো হয়নি)

5. Develop a short sequence of instructions that adds AL, BL, CL, DL, and AH. Save the sum in the DH register.

English:

ADD AH, AL

ADD AH, BL

ADD AH, CL

ADD AH, DL

MOV DH, AH

Bangla:

প্রথমে AH-তে AL যোগ করে, তারপর BL, CL, DL একে একে যোগ করে। শেষে AH-এর মান DH-তে রাখে। কারণ, সব যোগফল AH-তে জমা হবে, শেষে DH-তে কপি করলে উত্তর পাওয়া যাবে।

6. Develop a short sequence of instructions that adds AX, BX, CX, DX, and SP. Save the sum in the DL register.

English:

ADD AX, BX

ADD AX, CX

ADD AX, DX

ADD AX, SP

MOV DL, AL

Bangla:

সব যোগফল AX-তে জমা হবে। শেষে AX-এর সোয়ার কইট (AL) DL-এ কপি করে। কারণ, DL-এ শুধু ৮-বিট মান রাখা যায়।

---

7. Develop a short sequence of instructions that adds ECX, EDX, and ESI. Save the sum in the EDI register.

English:  
MOV EDI, ECX  
ADD EDI, EDX  
ADD EDI, ESI

Bangla:  
প্রথমে ECX-এর মান EDI-তে রাখে, তারপর EDX ও ESI যোগ করে। কারণ, সব যোগফল EDI-তে জমা হবে।

---

8. Develop a short sequence of instructions that adds RCX, RDX, and RSI. Save the sum in the R12 register.

English:  
MOV R12, RCX  
ADD R12, RDX  
ADD R12, RSI

Bangla:  
প্রথমে RCX-এর মান R12-তে রাখে, তারপর RDX ও RSI যোগ করে। কারণ, সব যোগফল R12-তে জমা হবে।

---

9. Select an instruction that adds BX to DX, and also adds the contents of the carry flag (C) to the result.

English: ADC DX, BX  
Bangla:

ADC (Add with Carry) নির্দেশ কারি ক্যারি ফ্ল্যাগসহ যোগফল দেয়। তাই, BX-কে DX-এ যোগ করে এবং কারি ফ্ল্যাগও যোগ হবে।

---

10. Choose an instruction that adds 1 to the contents of the SP register.

English: INC SP  
Bangla:

INC নির্দেশ কোনো রেজিস্টারের মান 1 বাড়ায়। তাই, SP-তে 1 যোগ করতে INC SP ব্যবহার করে।

---

---

11. What is wrong with the INC [BX] instruction?

উত্তর:

INC [BX] নির্দেশে কোন সাইজের ভেটা (৮, ১৬, ৩২, ৬৪ বিট) বাড়াতে হবে, সেটা স্পষ্ট নয়। তাই, BYTE PTR, WORD PTR ইত্যাদি দিয়ে স্পষ্ট করতে হয়।

উদাহরণ:

- ভুল: INC [BX]
- ঠিক: INC BYTE PTR [BX] (৮-বিট বাড়ানো)
- ঠিক: INC WORD PTR [BX] (১৬-বিট বাড়ানো)

ব্যাখ্যা:

কম্পিউটার বুঝতে পারে না, তুমি কত বিট বাড়াতে চাও। তাই স্পষ্টভাবে বলতে হয়, যাতে প্রসেসর জানে ঠিক কতটুকু ভেটা বাড়াতে হবে।

---

12. Select a SUB instruction that will:

(a) subtract BX from CX

উত্তর: SUB CX, BX

উদাহরণ:

CX = 10, BX = 3

SUB CX, BX → CX = 10 - 3 = 7

ব্যাখ্যা:

SUB নির্দেশে প্রথম রেজিস্টার থেকে দ্বিতীয় রেজিস্টার বাদ যায়। এখানে CX থেকে BX বাদ হবে।

---

(b) subtract 0EEH from DH

উত্তর: SUB DH, 0EEH

উদাহরণ:

DH = 100H

SUB DH, 0EEH → DH = 100H - 0EEH = 12H

ব্যাখ্যা:

এখানে DH রেজিস্টার থেকে 0EEH মানটি বাদ দেওয়া হচ্ছে।

---

---

(c) subtract DI from SI

উত্তর: `SUB SI, DI`

উদাহরণ:

SI = 50, DI = 20  
 $SUB\ SI, DI \rightarrow SI = 50 - 20 = 30$

ব্যাখ্যা:

SI থেকে DI বাদ দিলে SI-তে নতুন মান জমা হয়।

---

(d) subtract 3322H from EBP

উত্তর: `SUB EBP, 3322H`

উদাহরণ:

EBP = 5000H  
 $SUB\ EBP, 3322H \rightarrow EBP = 5000H - 3322H = 1CDEH$

ব্যাখ্যা:

EBP থেকে 3322H বাদ দিলে EBP-তে নতুন মান জমা হয়।

---

(e) subtract the data addressed by SI from CH

উত্তর: `SUB CH, [SI]`

উদাহরণ:

SI = 2000H, [2000H] = 5, CH = 10  
 $SUB\ CH, [SI] \rightarrow CH = 10 - 5 = 5$

ব্যাখ্যা:

SI খেই মেমরি লোকেশন দেখায়, সেই ভেটা CH থেকে বাদ দেওয়া হচ্ছে।

---

(f) subtract the data stored 10 words after the location addressed by SI from DX

উত্তর: `SUB DX, [SI+10]`

উদাহরণ:

SI = 1000H, [1014H] = 8, DX = 20  
 $SUB\ DX, [SI+10] \rightarrow DX = 20 - 8 = 12$

ব্যাখ্যা:

SI থেকে ১০ ওয়ার্ড পরে খেই মেমরি লোকেশন, সেই ভেটা DX থেকে বাদ দেওয়া হচ্ছে।

---

(g) subtract AL from memory location FROG

উত্তর: `SUB FROG, AL`

উদাহরণ:

FROG = 15, AL = 5  
 $SUB\ FROG, AL \rightarrow FROG = 15 - 5 = 10$

ব্যাখ্যা:

FROG নামের মেমরি লোকেশন থেকে AL-এর মান বাদ দেওয়া হচ্ছে।

---

(h) subtract R9 from R10

উত্তর: `SUB R10, R9`

উদাহরণ:

R10 = 100, R9 = 30  
 $SUB\ R10, R9 \rightarrow R10 = 100 - 30 = 70$

ব্যাখ্যা:

R10 থেকে R9 বাদ দিলে R10-এ নতুন মান জমা হয়।

---

13. If DL = 0F3H and BH = 72H, list the difference after BH is subtracted from DL and show the contents of the flag register bits.

উত্তর:

DL = 81H, S = 1, Z = 0, C = 0, A = 0, P = 0, O = 1

উদাহরণ:

DL = 0F3H (243), BH = 72H (114)  
 $SUB\ DL, BH \rightarrow DL = 243 - 114 = 129 = 81H$   
Sign (S) = 1 (কারণ ৮১H-এর সবচেয়ে বাম বিট ১, যানে নেগেটিভ)  
Zero (Z) = 0 (ফলাফল শূন্য নয়)  
Carry (C) = 0 (ক্যারি হয়নি)  
Overflow (O) = 1 (ওভারফ্লো হয়েছে)

ব্যাখ্যা:

ফলাফল নেগেটিভ হলে S=1, শূন্য হলে Z=1, ক্যারি হলে C=1, ওভারফ্লো হলে O=1 হয়।

---

**14. Write a short sequence of instructions that subtracts the numbers in DI, SI, and BP from the AX register. Store the difference in register BX.**

উত্তর:

```
MOV BX, AX
SUB BX, DI
SUB BX, SI
SUB BX, BP
```

উদাহরণ:

```
AX = 50, DI = 10, SI = 5, BP = 2
MOV BX, AX → BX = 50
SUB BX, DI → BX = 50 - 10 = 40
SUB BX, SI → BX = 40 - 5 = 35
SUB BX, BP → BX = 35 - 2 = 33
```

ব্যাখ্যা:

প্রথমে AX-এর মান BX-তে রাখা, তারপর একে একে DI, SI, BP বাদ দাও। সবশেষে BX-তে চূড়ান্ত মান থাকবে।

**15. Choose an instruction that subtracts 1 from register EBX.**

উত্তর: DEC EBX

উদাহরণ:

```
EBX = 10
DEC EBX → EBX = 9
```

ব্যাখ্যা:

DEC যানে ১ কমানো। তাই DEC EBX দিলে EBX-এর মান ১ কমে যাবে।

**16. Explain what the SBB [DI-4], DX instruction accomplishes.**

উত্তর:

[DI-4] লোকেশনের ভেতর থেকে DX এবং ক্যারি স্ট্যাগ বাদ দেওয়া হবে, এবং ফলাফল DX-তে রাখা হবে।

উদাহরণ:

```
[DI-4] = 20, DX = 5, Carry = 1
SBB DX, [DI-4] → DX = 20 - 5 - 1 = 14
```

ব্যাখ্যা:

SBB যানে Subtract with Borrow (Carry)। Carry থাকলে আরও ১ বাদ যাবে।

**17. Explain the difference between the SUB and CMP instruction.**

উত্তর:

SUB যানে বাদ দেওয়া এবং ফলাফল রেখে দেওয়া।

CMP যানে শুধু তুলনা করা, কোনো মান বদলায় না, শুধু ফ্লাগ সেট হয়।

উদাহরণ:

```
AX = 10, BX = 5
SUB AX, BX → AX = 5
CMP AX, BX → AX = 10 (কিছুই বদলায় না, শুধু ফ্লাগ সেট হয়)
```

ব্যাখ্যা:

SUB রেজিস্টারের মান বদলায়, CMP শুধু তুলনা করে ফ্লাগ সেট করে।

**18. When two 8-bit numbers are multiplied, where is the product found?**

উত্তর:

AH (most significant) এবং AL (least significant)

উদাহরণ:

```
AL = 10, BL = 20
MUL BL → 10 × 20 = 200
AH:AL = 00C8H (AH = 00, AL = C8)
```

ব্যাখ্যা:

৮-বিট × ৮-বিট = ১৬-বিট ফলাফল। উচ্চ ৮-বিট AH-তে, নিচের ৮-বিট AL-এ থাকে।

**19. When two 16-bit numbers are multiplied, what two registers hold the product?**

উত্তর:

DX (most significant), AX (least significant)

উদাহরণ:

```
AX = 1000H, BX = 0002H
MUL BX → 1000H × 2 = 2000H
DX:AX = 0000:2000H (DX = 0, AX = 2000H)
```

ব্যাখ্যা:

১৬-বিট × ১৬-বিট = ৩২-বিট ফলাফল। উচ্চ ১৬-বিট DX-তে, নিচের ১৬-বিট AX-এ থাকে।

---

**প্রশ্ন:**

When two numbers multiply, what happens to the O (Overflow) and C (Carry) flag bits?

---

**উত্তর ও ব্যাখ্যা:****১. গুণফল কোথায় রাখা হয়?**

- \* ৮-বিট \* ৮-বিট  $\rightarrow$  ১৬-বিট (ফলাফল AH:AL-এ)
- \* ১৬-বিট \* ১৬-বিট  $\rightarrow$  ৩২-বিট (ফলাফল DX:AX-এ)

**২. O (Overflow) এবং C (Carry) ফ্লাগ কবে ১ হয়?**

- \* যদি গুণফল পুরোপুরি নিচের রেজিস্টারে (AL বা AX) ধরে যায়, তাহলে O ও C ফ্লাগ ০ হয়।
  - \* যদি গুণফল নিচের রেজিস্টারে না ধরে, অর্থাৎ উপরের রেজিস্টারে (AH বা DX) কিছু থাকে, তাহলে O ও C ফ্লাগ ১ হয়।
- 

**৩. উদাহরণ ১: Overflow/Carry হবে না (ফ্লাগ = ০)**

ধরা যাক:

AL = 10 (0AH), BL = 20 (14H)  
MUL BL  $\rightarrow$  10 \* 20 = 200 (C8H)

- \* ২০০ (C8H) ৮-বিট AL-এ ধরে যায়, AH = 0
  - \* তাই, O = 0, C = 0
- 

**৪. উদাহরণ ২: Overflow/Carry হবে (ফ্লাগ = ১)**

ধরা যাক:

AL = 200 (C8H), BL = 2  
MUL BL  $\rightarrow$  200 \* 2 = 400 (190H)

- \* ৪০০ (190H) ১৬-বিট লগে:
    - \* AL = 90H (নিচের ৮-বিট)
    - \* AH = 01H (উপরের ৮-বিট)
  - \* এখানে AH  $\neq$  0, যাবে AL-এ পুরো ফলাফল ধরে নাই।
  - \* তাই, O = 1, C = 1
- 

**৫. সংক্ষেপে মনে রাখো:**

- \* ফলাফল AL/AX-এ পুরোটা ধরে গেলে:  
O = 0, C = 0
- \* ফলাফল AL/AX-এ না ধরে, AH/DX-এ কিছু থাকলে:  
O = 1, C = 1

---

**21. Where is the product stored for the MUL EDI instruction?**

Answer: EDI and EAX as a 64-bit product.

উদাহরণ ও ব্যাখ্যা:

যখন ৩২-বিট EDI রেজিস্টারকে গুণ করা হয়, তখন ফলাফল ৬৪-বিট হয়।

- নিয়ে ৩২-বিট EAX-এ
  - উপরে ৩২-বিট EDI-এ
- যেমন,  $EDI \times কিছু \rightarrow$  ফলাফল EDI:EAX-এ জায়গা হয়।
- 

**22. Write a sequence of instructions that cube the 8-bit number found in DL. Load DL with a 5 initially, and make sure that your result is a 16-bit number.**

Answer:

```
MOV DL,5
MOV AL,DL
MUL DL
MUL DL
```

উদাহরণ ও ব্যাখ্যা:

- প্রথমে DL-এ ৫ আছে
  - AL-এ DL-এর মান আছে
  - প্রথম MUL DL  $\rightarrow AL \times DL = ৫ \times ৫ = ২৫$
  - দ্বিতীয় MUL DL  $\rightarrow$  আগের ফলাফল  $\times DL = ২৫ \times ৫ = ১২৫$
  - ফলাফল ১৬-বিট (AH:AL-এ)
- 

**23. What is the difference between the IMUL and MUL instructions?**

Answer: IMUL is signed multiplication while MUL is unsigned.

উদাহরণ ও ব্যাখ্যা:

- MUL: শুধু বিনামূল্য (unsigned) সংখ্যা গুণ করে
  - IMUL: বিনামূল্য ও স্বাক্ষর (signed) সংখ্যা গুণ করতে পারে
- যেমন,  $-5 \times 2 \rightarrow$  IMUL দিয়ে হবে, MUL দিয়ে হবে না।
- 

**24. Describe the operation of the IMUL BX,DX,100H instruction.**

Answer:  $BX = DX \text{ times } 100H$

উদাহরণ ও ব্যাখ্যা:

- IMUL BX, DX, 100H
- যাৎ,  $BX = DX \times 100H$
- যদি  $DX = 2$ , তাহলে  $BX = 2 \times 256 = 512$

25. When 8-bit numbers are divided, in which register is the dividend found?

Answer: AX

Example & Explanation:

ধরা যাক, দু'টি ৮-বিট সংখ্যা ভাগ করতে চাই।

- ধরে, ভাগফল বের করতে হবে  $২০ \div ৪$

কোথায় রাখতে হবে?

- AL-এ রাখতে হবে ভাগফল, AH-তে থাকবে ভাগশেষ (remainder)।
- কিন্তু ভাগ করার সময়, পুরো ১৬-বিট AX রেজিস্টার ব্যবহার হয় (যদিও শুধু AL-এ ভেটী রাখলেও হয়)।

কোড:

assembly

MOV AL, 20 ; AL = 20 (ভাগফল এখনে রাখবে)  
MOV AH, 0 ; AH = 0 (ভাগশেষ এখনে রাখবে)  
MOV BL, 4 ; BL = 4 (ভাগক)  
DIV BL ; AL ← BL

- DIV BL দিলে,  $২০ \div ৪ = ৫$
- AL = ৫ (ভাগফল), AH = ০ (ভাগশেষ)

কেন AX?

কারণ, DIV ইনস্ট্রাকশন ৮-বিট ভাগ করার সময় AX (AL+AH) রেজিস্টারকে dividend হিসেবে ধরে।

26. When 16-bit numbers are divided, in which register is the quotient found?

Answer: AX

Example & Explanation:

ধরা যাক, দু'টি ১৬-বিট সংখ্যা ভাগ করতে চাই:

- ধরে,  $১০০০ \div ১০ = ১০০$

কোথায় রাখতে হবে?

- ভাগফল AX-এ, ভাগশেষ DX-তে

কোড:

assembly

MOV AX, 1000 ; AX = 1000 (গুণ্য সংখ্যা)  
MOV DX, 0 ; DX = 0 (১৬-বিট, এখনে ০)  
MOV BX, 10 ; BX = 10 (ভাগক)  
DIV BX ; DX:AX ← BX

```
assembly
MOV AX, 1000 ; AX = 1000 (পঞ্চা সহস্র)
MOV DX, 0 ; DX = 0 (দুই ১৬-বিট, এরমধ্যে ০)
MOV BX, 10 ; BX = 10 (দশক)
DIV BX ; DX:AX = BX
```

- $DIV\ BX$  দিলে,  $1000 \div 10 = 100$
- $AX = 100$  (ভাগফল),  $DX = 0$  (ভাগশেষ)

কেন AX?

কারণ, ১৬-বিট ভাগফল সবসময় AX-তে পাওয়া যায়।

## 27. When 64-bit numbers are divided, in which register is the quotient found?

Answer: RAX

### Example & Explanation:

ধরা যাক, দু'মি ৬৪-বিট সংখ্যা ভাগ করতে চাই:

- ধরো,  $100000000000$  (১২ ডিজিট)  $\div 1000 = 1000000000$

কেন যায় রাখতে হবে?

- ভাগফল RAX-এ, ভাগশেষ RDX-তে

কোড:

```
assembly
MOV RAX, 100000000000 ; RAX = ১০ সহস্র (১৬-বিট)
MOV RDX, 0 ; RDX = 0 (দুই ৬৪-বিট)
MOV RCX, 1000 ; RCX = 1000 (দশক)
DIV RCX ; RDX:RAX = RCX
```

- $DIV\ RCX$  দিলে,  $100000000000 \div 1000 = 1000000000$
- $RAX = 1000000000$  (ভাগফল),  $RDX = 0$  (ভাগশেষ)

কেন RAX?

কারণ, ৬৪-বিট ভাগফল সবসময় RAX-এ পাওয়া যায়।

### সংক্ষেপে মনে রাখো:

- ৮-বিট ভাগফল AL-এ, ভাগশেষ AH-তে (কিন্তু dividend AX-তে রাখতে হয়)
- ১৬-বিট ভাগফল AX-এ, ভাগশেষ DX-তে (dividend DX:AX-তে রাখতে হয়)
- ৬৪-বিট ভাগফল RAX-এ, ভাগশেষ RDX-তে (dividend RDX:RAX-তে রাখতে হয়)

## 28. What errors are detected during a division?

Answer: Divide overflow and divide by zero.

### Example 1: Divide by zero

- ধরো,  $AL = 15$ ,  $BL = 0$
- কোড:

```
text
MOV AL, 15
MOV BL, 0
DIV BL
```

- এখানে  $BL = 0$ , মানে ভাগক শূন্য।
- $15 \div 0$  করা যায় না, তাই **divide by zero error** হবে।
- কেন?: কোনো সংখ্যাকে শূন্য দিয়ে ভাগ করা যায় না, তাই প্রসেসর error দেখায়।

### Example 2: Divide overflow

- ধরো,  $AL = 255$ ,  $BL = 2$
- কোড:

```
text
MOV AL, 255
MOV BL, 2
DIV BL
```

- $255 \div 2 = 127.5$
- ভাগফল AL-এ ৮-বিট ধরে, কিন্তু যদি ভাগফল AL-এ না ধরে (যেমন, AL-এ ৮-বিটের বেশি হয়ে যায়), তাহলে **divide overflow error** হয়।
- কেন?: AL-এ ৮-বিটের বেশি মান রাখা যায় না, তাই overflow হয়।



## 29. Explain the difference between the IDIV and DIV instructions.

Answer: IDIV is signed division, while DIV is unsigned division.

### Example:

- DIV (Unsigned):  
AL = 10, BL = 2

```
text
MOV AL, 10
MOV BL, 2
DIV BL
```

$10 \div 2 = 5$  (AL-এ ৫ আসবে)  
এখানে শুধু ধনাত্মক সংখ্যা ভাগ করা যায়।

- IDIV (Signed):  
AL = -10 (signed), BL = 2

```
text
MOV AL, -10
MOV BL, 2
IDIV BL
```

$-10 \div 2 = -5$  (AL-এ -৫ আসবে)  
এখানে ধনাত্মক ও ঋণাত্মক দুই ধরনের সংখ্যা ভাগ করা যায়।

### কেন?

- DIV শুধু unsigned (positive) সংখ্যা ভাগ করে।
- IDIV signed (positive/negative) সংখ্যা ভাগ করতে পারে।

## 30. Where is the remainder found after an 8-bit division?

Answer: AH

### Example:

- AL = 23, BL = 5
- কোড:

```
text
MOV AL, 23
MOV AH, 0
MOV BL, 5
DIV BL
```

- $23 \div 5 = 4$  (ভাগফল AL-এ), ভাগশেষ ৩ (AH-তে)
- কেন?: ৮-বিট ভাগ করার পর, ভাগফল AL-এ, ভাগশেষ AH-তে জমা হয়।

## 31. Where is the quotient found after a 64-bit division?

Answer: RAX

### Example:

- ধরে, RDX:RAX = 10000000000 (একটি বড় ৬৪-বিট সংখ্যা)
- RCX = 1000
- কোড:

```
text
MOV RAX, 10000000000
MOV RDX, 0
MOV RCX, 1000
DIV RCX
```

- ভাগফল RAX-এ, ভাগশেষ RDX-তে
- $10000000000 \div 1000 = 10000000$  (RAX-এ), ভাগশেষ ০ (RDX-তে)
- কেন?: ৬৪-বিট ভাগফল সবসময় RAX-এ পাওয়া যায়, কারণ RAX ৬৪-বিট রেজিস্টার।

### প্রশ্ন:

BL-এ থাকা সংখ্যাকে CL-এ থাকা সংখ্যায় ভাগ করে, তারপর ফলাফলকে ২ দিয়ে গুণ করে। চূড়ান্ত ফলাফল ১৬-বিট DX রেজিস্টারে রাখতে হবে।

### Step by Step Example:

#### ধরা যাক:

BL = 10

CL = 2

#### ১. MOV AH, 0

- \* AH = 0
- \* কারণ, ৮-বিট ভাগ করার সময় Div ইন্সট্রাকশন dividend হিসেবে AX (যেহেতু AH:AL) নেয়।
- \* এখানে AL-এ BL-এর মান যাবে, AH-তে ০ রাখতে হয়।

#### ২. MOV AL, BL

- \* AL = BL = 10
- \* এখন AX = 000A (AH = 0, AL = 10)

#### ৩. DIV CL

- \*  $DIV\ CL \rightarrow AX \div CL$
- \* অর্থাৎ,  $10 \div 2 = 5$
- \* ভাগফল AL-এ (AL = 5), ভাগশেষ AH-তে (AH = 0)

#### ৪. ADD AL, AL

- \*  $AL = AL + AL = 5 + 5 = 10$
- \* এখন AL = 10
- \* (এটা যেনে, ভাগফলকে ২ দিয়ে গুণ করা)

#### ৫. MOV DL, AL

- \* DL = AL = 10
- \* এখন DL-এ চূড়ান্ত ফলাফল আছে

#### ৬. MOV DH, 0

- \* DH = 0
- \* কারণ, DX রেজিস্টার ১৬-বিট, তার উচ্চ ৮-বিটে (DH) ০ রাখতে হবে

#### ৭. ADC DH, 0

- \* যদি আগের কোনো অপারেশনে ক্যারি (carry) হয়ে থাকে, তাহলে সেটা DH-তে যোগ হবে
- \* এখানে কোনো ক্যারি নেই, তাই DH = ০.ই থাকবে

### শেষে:

- \*  $DX = DH:DL = 000A$  (Hexadecimal) = 10 (Decimal)

### সংক্ষেপে:

১. BL-এর মান AL-এ রাখা, AH-এ ০ রাখা
২. CL দিয়ে ভাগ করা (DIV CL)
৩. ভাগফলকে ২ দিয়ে গুণ করা (ADD AL, AL)
৪. DL-এ ফলাফল রাখা, DH-এ ০ রাখা
৫. DX-তে চূড়ান্ত ১৬-বিট ফলাফল পড়ানো যাবে

### আরও সহজে মনে রাখা:

- \*  $BL \rightarrow AL$
- \* AH = 0
- \*  $DIV\ CL \rightarrow AL = (BL \div CL)$
- \*  $AL \times 2$
- \* DX = ১৬-বিট ফলাফল

## 35. Which instructions are used with ASCII arithmetic operations?

Answer: AAA, AAS, AAD, and AAM

### উদাহরণ ও ব্যাখ্যা:

- \* AAA (ASCII Adjust after Addition): যোগফলকে ASCII ফরম্যাটে ঠিক করে।
- \* AAS (ASCII Adjust after Subtraction): বিয়োগফলকে ASCII ফরম্যাটে ঠিক করে।
- \* AAD (ASCII Adjust before Division): ভাগ করার আগে ASCII ফরম্যাটে ঠিক করে।
- \* AAM (ASCII Adjust after Multiplication): গুণফলকে ASCII ফরম্যাটে ঠিক করে।

যখন তুমি ASCII সংখ্যার উপর গাণিতিক অপারেশন করো, তখন এই নির্দেশগুলো ব্যবহার করতে হয়।

৩৬. **AX-এ থাকা সংখ্যা ৫ ডিজিট BCD-তে কনভার্ট করে মেমরিতে রাখার জন্য নির্দেশাবলি ও উদাহরণ**

Algorithm:

text

```
PUSH DX
PUSH CX

MOV CX,10000
DIV CX
MOV [BX],AL
MOV AL,DX
POP CX

MOV CX,1000
DIV CX
MOV [BX+1],AL
MOV AL,DX
POP CX

MOV CX,100
DIV CX
MOV [BX+2],AL
MOV AL,DX
POP CX

MOV CX,10
DIV CX
MOV [BX+3],AL
MOV AL,DX
POP CX

MOV [BX+4],AL
POP DX
```

Note (Example with AX = 54321, BX = 2000H):

1. MOV CX,10000

\* এখন CX = 10000

2. DIV CX

\*  $54321 \div 10000 = 5$  (AL-এ), বাকি 4321 (DX-তে)  
\*  $[BX] = 5 \rightarrow [2000H] = 5$

3. MOV AL,DX

\* AL = 4321 (বাকি অংশ)

4. MOV CX,1000

\* CX = 1000

4. MOV CX,1000

\* CX = 1000

5. DIV CX

\*  $4321 \div 1000 = 4$  (AL-এ), বাকি 321 (DX-তে)  
\*  $[BX+1] = 4 \rightarrow [2001H] = 4$

6. MOV AL,DX

\* AL = 321

7. MOV CX,100

\* CX = 100

8. DIV CX

\*  $321 \div 100 = 3$  (AL-এ), বাকি 21 (DX-তে)  
\*  $[BX+2] = 3 \rightarrow [2002H] = 3$

9. MOV AL,DX

\* AL = 21

10. MOV CX,10

\* CX = 10

11. DIV CX

\*  $21 \div 10 = 2$  (AL-এ), বাকি 1 (DX-তে)  
\*  $[BX+3] = 2 \rightarrow [2003H] = 2$

12. MOV AL,DX

\* AL = 1

13. MOV [BX+4],AL

\*  $[2004H] = 1$

ফলাফল:

[2000H] = 5

[2001H] = 4

[2002H] = 3

[2003H] = 2

[2004H] = 1

(অর্থাৎ, 54321 সংখ্যার প্রতিটি ডিজিট ফলাফল আলাদা মেমরিতে BCD আকারে সংরক্ষিত)

৩৭. **AX ও BX-তে থাকা ৮-ডিজিট BCD এবং CX ও DX-তে থাকা ৮-ডিজিট BCD যোগ করার জন্য নির্দেশাবলি ও উদাহরণ**

Algorithm:

text

```
PUSH AX
MOV AL, BL
ADD AL, DL
DAA
MOV AL, BH
ADC AL, DH
DAA
MOV BX, AX
POP AX
ADC AL, CL
```

**Note (Example with AX=1234, BX=5678, CX=1111, DX=2222):**

1. MOV AL, BL
  - BL = 78 (BX-এর নিম্ন ২ ডিজিট), AL = 78
2. ADD AL, DL
  - DL = 22 (DX-এর নিম্ন ২ ডিজিট), AL = 78 + 22 = 100
3. DAA
  - AL = 100 → BCD রীতি করলে AL = 00, ক্যারি ১
4. MOV AL, BH
  - BH = 56 (BX-এর উচ্চ ২ ডিজিট), AL = 56
5. ADC AL, DH
  - DH = 22 (DX-এর উচ্চ ২ ডিজিট), আগের ক্যারি ১
  - AL = 56 + 22 + 1 = 79
6. DAA
  - AL = 79 (BCD রীতি, কোনো ক্যারি নেই)
7. MOV BX, AX
  - BX-তে এখন নিম্ন ৪ ডিজিটের যোগফল (7900) আছে
8. POP AX
  - আগের AX ফেরত আনো (1234)
  - $DA = 1234$  এর জন্য  $17H$  ও  $12H$  অক্ষর ব্যবহার করা হয়। (1234) আছে
8. POP AX
  - আগের AX ফেরত আনো (1234)
9. ADC AL, CL
  - CL = 11 (CX-এর নিম্ন ২ ডিজিট), AL = AL + CL + আগের ক্যারি

ফলাফল:

- BX = 7900 (নিম্ন ৪ ডিজিট)
- AX = 2345 (উচ্চ ৪ ডিজিট)  
(অর্থাৎ,  $12345678 + 11112222 = 23457900$ )

**38. Does the AAM instruction function in the 64 bit mode?**

Answer: Neither the BCD or the ASCII instructions function in the 64 bit mode.

উদ্ভাৱণ ও ব্যাখ্যা:

৬৪-বিট প্রসেসরে AAM, AAD, AAA, AAS ইত্যাদি ASCII/BCD নির্দেশ কাজ করে না।  
কারণ, এগুলো শুধুমাত্র ৮/১৬/৩২-বিট প্রসেসরের জন্য ডিজাইন করা।

**39. Select an AND instruction that will:**

**(a) AND BX with DX and save the result in BX**

Answer: AND BX, DX

উদ্ভাৱণ:

BX = 1100, DX = 1010

AND BX, DX  $\rightarrow$  BX = 1000

**(b) AND 0EAH with DH**

Answer: AND DH, 0EAH

উদ্ভাৱণ:

DH = 1111 0000, 0EAH = 1110 1010

AND DH, 0EAH  $\rightarrow$  DH = 1110 0000

**(c) AND 1111 with BP and save the result in BP**

Answer: AND BP, 1111

উদ্ভাৱণ:

BP = 1011 1100, 1111 = 0000 1111

AND BP, 1111  $\rightarrow$  BP = 0000 1100

**(d) AND 1122H with EAX and save the result in EAX**

Answer: AND EAX, 1122H

উদ্ভাৱণ:

EAX = 1234H, 1122H = 0001 0001 0010 0010

AND EAX, 1122H  $\rightarrow$  EAX = 0001 0000 0010 0000

**(e) AND the data addressed by BP with CX and save the result in memory - AND [BP],CX**

Answer: AND [BP], CX

উদ্ভাৱণ:

[BP] = 1100, CX = 1010

AND [BP], CX  $\rightarrow$  [BP] = 1000

**(f) AND the data stored in four words before the location addressed by SI with DX and save the result in DX AND DX, [SI-8]**

Answer: AND DX, [SI-8]

উদ্ভাৱণ:

[SI-8] = 1111 0000, DX = 1010 1010

AND DX, [SI-8]  $\rightarrow$  DX = 1010 0000

**(g) AND AL with memory location WHAT and save the result at location WHAT**

Answer: AND WHAT, AL

উদ্ভাৱণ:

AL = 1100, WHAT = 1010

AND WHAT, AL  $\rightarrow$  WHAT = 1000

**40. Develop a short sequence of instructions that clears (0) the three leftmost bits of DH without changing the remainder of DH and stores the result in BH.**

Answer:

MOV BH, DH

AND BH, 1FH

উদ্ভাৱণ ও ব্যাখ্যা:

ধরা যাক, DH = 1111 0110

• MOV BH, DH  $\rightarrow$  BH = 1111 0110

• AND BH, 1FH  $\rightarrow$  1FH = 0001 1111

• BH = 1111 0110 AND 0001 1111 = 0001 0110

• অর্থাৎ, বাম দিকের ৩টি বিট ০ হয়ে গেল, বাকি অংশ অপরিবর্তিত থাকল।

কেন?

AND অপারেশন করলে নির্দিষ্ট বিট ০ করা যায়, আর বাকি বিট অপরিবর্তিত থাকে।

41. Select an OR instruction that will:

(a) OR BL with AH and save the result in AH

Answer: OR AH, BL

Example:

- AH = 1010 (binary), BL = 1100
- OR AH, BL  $\rightarrow$  AH = 1010 OR 1100 = 1110

ব্যাখ্যা:

OR করলে দুই বিটের যেকোনো একটিতে ১ থাকলে ফলাফল ১ হয়।

---

(b) OR 88H with ECX

Answer: OR ECX, 88H

Example:

- ECX = 0000 0101, 88H = 1000 1000
- OR ECX, 88H  $\rightarrow$  ECX = 1000 1101

ব্যাখ্যা:

ECX-তে ৮৮H যোগ করলে যেসব বিটে ১ আছে, সেগুলো ১ হয়ে যাবে।

---

(c) OR DX with SI and save the result in SI

Answer: OR SI, DX

Example:

- SI = 0101, DX = 0011
- OR SI, DX  $\rightarrow$  SI = 0111

ব্যাখ্যা:

SI-তে DX-এর সাথে OR করলে, যেসব বিটে যেকোনো একটিতে ১ আছে, সেগুলো ১ হবে।

---

(d) OR 1122H with BP

Answer: OR BP, 1122H

Example:

- BP = 1000, 1122H = 0001 0001 0010 0010
  - OR BP, 1122H  $\rightarrow$  BP-তে যেসব বিটে ১ আছে, সেগুলো ১ হয়ে যাবে।
- 

(e) OR the data addressed by RBX with RCX and save the result in memory

Answer: OR [RBX], RCX

Example:

- [RBX] = 0101, RCX = 0011
  - OR [RBX], RCX  $\rightarrow$  [RBX] = 0111
- 

(f) OR the data stored 40 bytes after the location addressed by BP with AL and save the result in AL

Answer: OR AL, [BP+40]

Example:

- [BP+40] = 1000, AL = 0100
  - OR AL, [BP+40]  $\rightarrow$  AL = 1100
- 

(g) OR AH with memory location WHEN and save the result in WHEN

Answer: OR WHEN, AH

Example:

- WHEN = 0011, AH = 0101
  - OR WHEN, AH  $\rightarrow$  WHEN = 0111
- 

42. Develop a short sequence of instructions that sets (1) the rightmost 5 bits of DI without changing the remaining bits of DI. Save the results in SI.

Answer:

MOV SI, DI

OR SI, 1FH

Example:

- DI = 1010 0000
- 1FH = 0001 1111
- OR SI, 1FH  $\rightarrow$  SI = 1010 0000 OR 0001 1111 = 1010 1111

ব্যাখ্যা:

OR করলে ডান দিকের ৫টি বিট সরাসরি ১ হয়ে যাবে, বাকি বিট অপরিবর্তিত থাকবে।

43. Select the XOR instruction that will:

(a) XOR BH with AH and save the result in AH

Answer: XOR AH, BH

Example:

- AH = 1100, BH = 1010
- XOR AH, BH  $\rightarrow$  AH = 0110

ব্যাখ্যা:

XOR করলে দুই বিট আলাদা হলে ১, একই হলে ০ হয়।

(b) XOR 99H with CL

Answer: XOR CL, 99H

Example:

- CL = 1001 0001, 99H = 1001 1001
- XOR CL, 99H  $\rightarrow$  CL = 0000 1000

(c) XOR DX with DI and save the result in DX

Answer: XOR DX, DI

Example:

- DX = 1100, DI = 1010
- XOR DX, DI  $\rightarrow$  DX = 0110

(d) XOR 1A23H with RSP

Answer: XOR RSP, 1A23H

Example:

- RSP = 1111 0000, 1A23H = 0001 1010 0010 0011
- XOR RSP, 1A23H  $\rightarrow$  RSP-তে বিটভেদে XOR হবে

(e) XOR the data addressed by EBX with DX and save the result in memory

Answer: XOR [EBX], DX

Example:

- [EBX] = 1100, DX = 1010
- XOR [EBX], DX  $\rightarrow$  [EBX] = 0110

(f) XOR the data stored 30 words after the location addressed by BP with DI and save the result in DI

Answer: XOR DI, [BP+60]

Example:

- [BP+60] = 1010, DI = 1100
- XOR DI, [BP+60]  $\rightarrow$  DI = 0110

(g) XOR DI with memory location WELL and save the result in DI

Answer: XOR DI, WELL

Example:

- WELL = 1001, DI = 0110
- XOR DI, WELL  $\rightarrow$  DI = 1111

44. Develop a sequence of instructions that sets (1) the rightmost 4 bits of AX; clears (0) the leftmost 3 bits of AX; and inverts bits 7, 8, and 9 of AX.

Answer:

OR AX, 0F

AND AX, 1FFFH

XOR AX, 03B0H

Example & Explanation:

- OR AX, 0F  $\rightarrow$  ডান দিকের ৪ বিট ১ হয়ে যাবে
- AND AX, 1FFFH  $\rightarrow$  বাম দিকের ৩ বিট ০ হয়ে যাবে
- XOR AX, 03B0H  $\rightarrow$  ৭, ৮, ৯ নম্বর বিট উল্টে যাবে। যদি ১ থাকে ০ হবে, ০ থাকলে ১ হবে।

45. Describe the difference between the AND and TEST instructions.

Answer:

AND changes the value of the logical product, but TEST only checks the bits and sets the flag, value পরিবর্তন করে না।

Example:

- AND AX, BX  $\rightarrow$  AX-এর মান বদলে যায়
- TEST AX, BX  $\rightarrow$  AX-এর মান বদলায় না, শুধু ফ্লাগ সেট হয়





### 33. Which instructions are used with BCD arithmetic operations?

Answer: DAA and DAS

Example & Explanation:

- DAA: যোগফলকে BCD ফরম্যাটে ঠিক করে
  - DAS: বিয়োগফলকে BCD ফরম্যাটে ঠিক করে  
যেমন, AL = 09, ADD AL, 01  $\rightarrow$  AL = 0A (BCD নয়), DAA দিলে AL = 10 (BCD)
- 

### 34. Explain how the AAM instruction converts from binary to BCD.

Answer: It divides AL by 10. This causes numbers between 0 and 99 decimal to be converted to unpacked BCD in AH (quotient) and AL (remainder).

Example & Explanation:

- AL = 45
- AAM দিলে, AL-কে ১০ দিয়ে ভাগ করবে
- AH = ৪ (কতবার ১০ দিয়ে ভাগ যায়), AL = ৫ (ভাগশেষ)
- অর্থাৎ, ৪৫  $\rightarrow$  AH = ৪, AL = ৫ (BCD ফরম্যাট)