operand

operand → Reg.

Memory

Constant

ADD R_0, R_1

ADD [100], R_1

[illegible]

ممنوع هيذ

ADD [100], [101]

انه الثنين ميموري

الحل

mov R_0, [100]

mov R_1, [101]

ADD R_2, R_1

mov [100], R1

bits of operand

get # of cells

Exp: 25 bit operand

25

then 2 cell

APP R0, 5

اكبر ا سي والرقم
بيكون مش مخزن
فاشي

وممنوع الرقم يكون
فالاول مثلا :

ADD 5, R_2

PS :

'1 operands is

registers or memory

then both have

same # of bits

---

when have constent

it must can rep.

Exp: constent 5

is 101 then at

least most value

3bit

---

unsigned int

$0 \rightarrow 2^n - 1$

signed (2's comp.)

$-2^{n-1} \rightarrow 2^{n-1} - 1$

Exp: # of bits?

± 512

± 513

2^9 2^{10}

$2^{n-1} = 2^{19}$

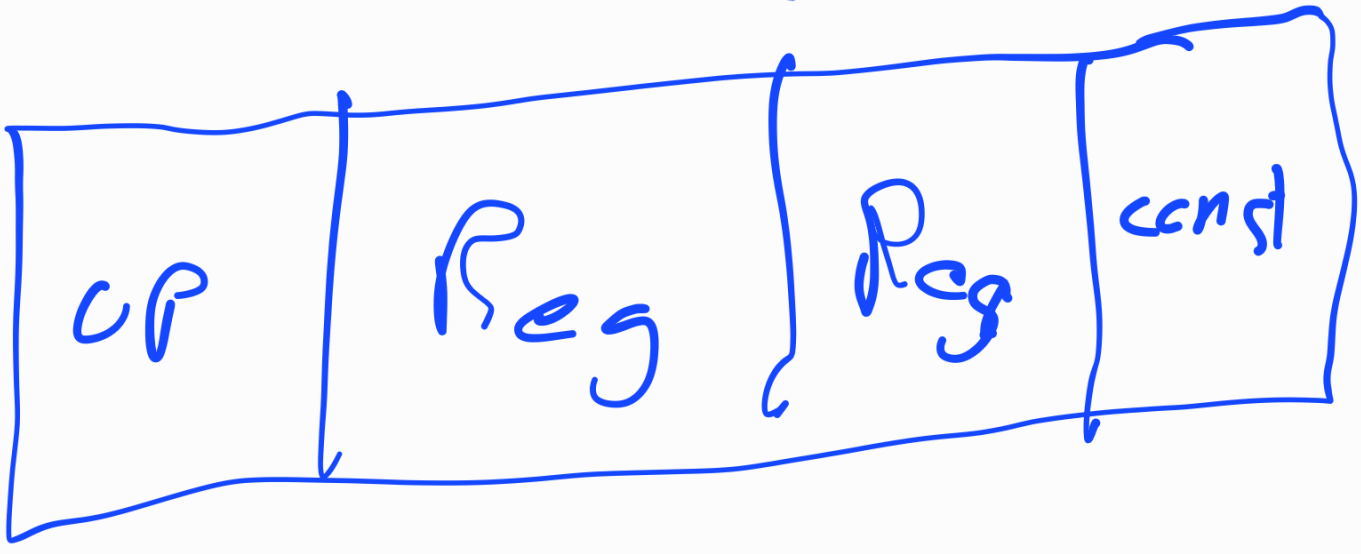
$\boxed{n = 20}$

64MB

2^6 2^{20}

bits = 26

32



48 operation

+ 8k constant

=> find # Reg

$$4 \text{ op} \leq 2^6$$

$$\text{op. code} = 6 \text{ bit}$$

$$8\text{K} \quad 2^3 \quad 2^{10}$$

$$2^{13} = 2^{n-1}$$

$$n = 14$$

$14 + 6 + 2R = 32$

$2R = 12$

$R = 6$ bit

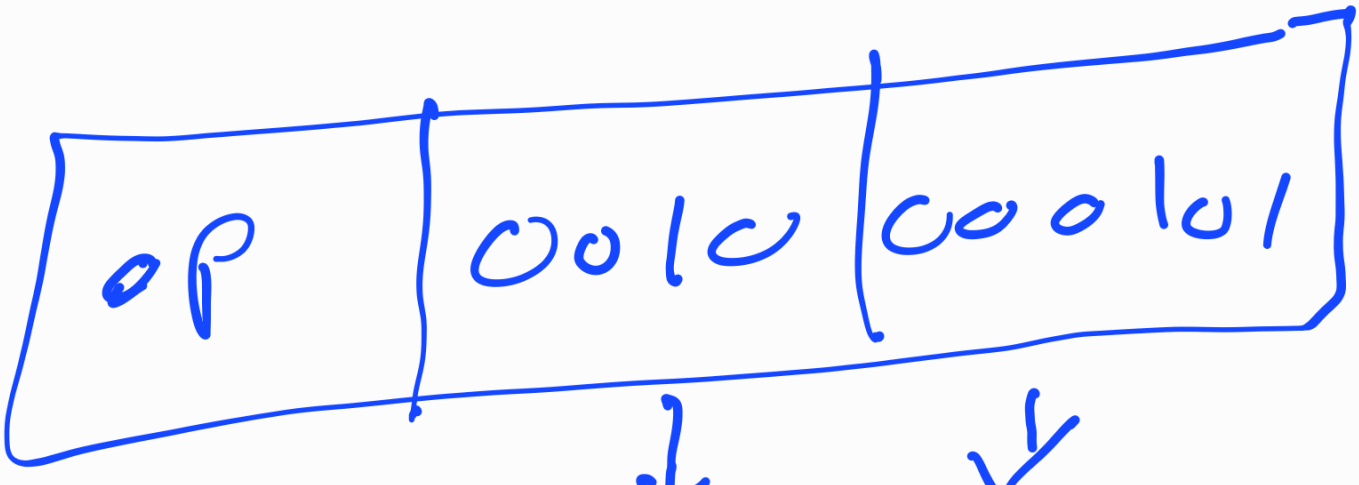
$2^6 = 64$ Reg

Dest. لـ Reg أول دائماً

ADD R_0, R_1, R_2

ADD R_0, R_1

للتخزين فيهم



ممكن تكون

Reg

Reg

memory

constant

كيف نحدد؟

1 نبرمج ال cpu

كل انه اشي ثابت

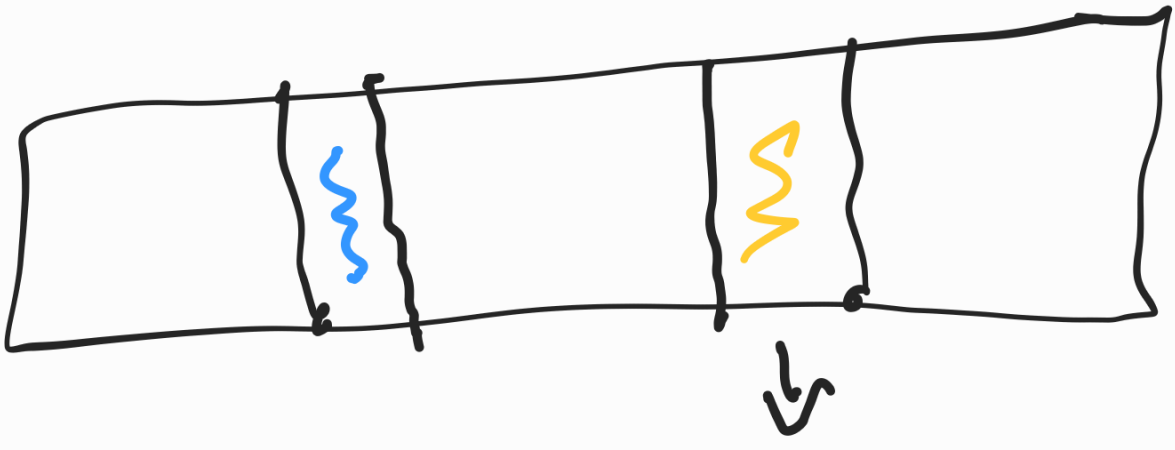
مثلاً نحدد انه أول

operand Reg

وثاني constant

For 2 bit

بيفهم يحدد نوعها

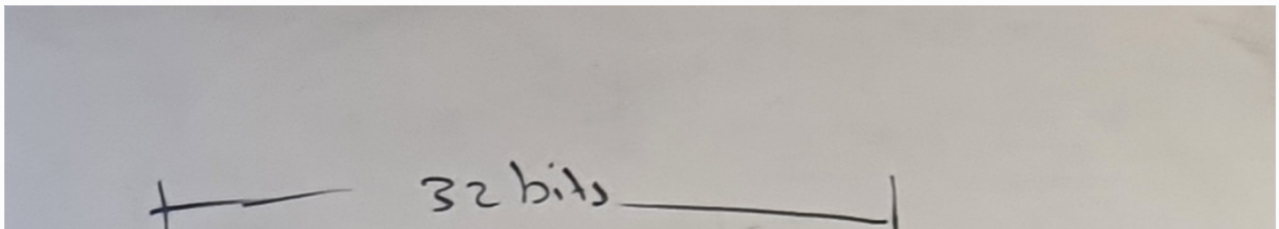


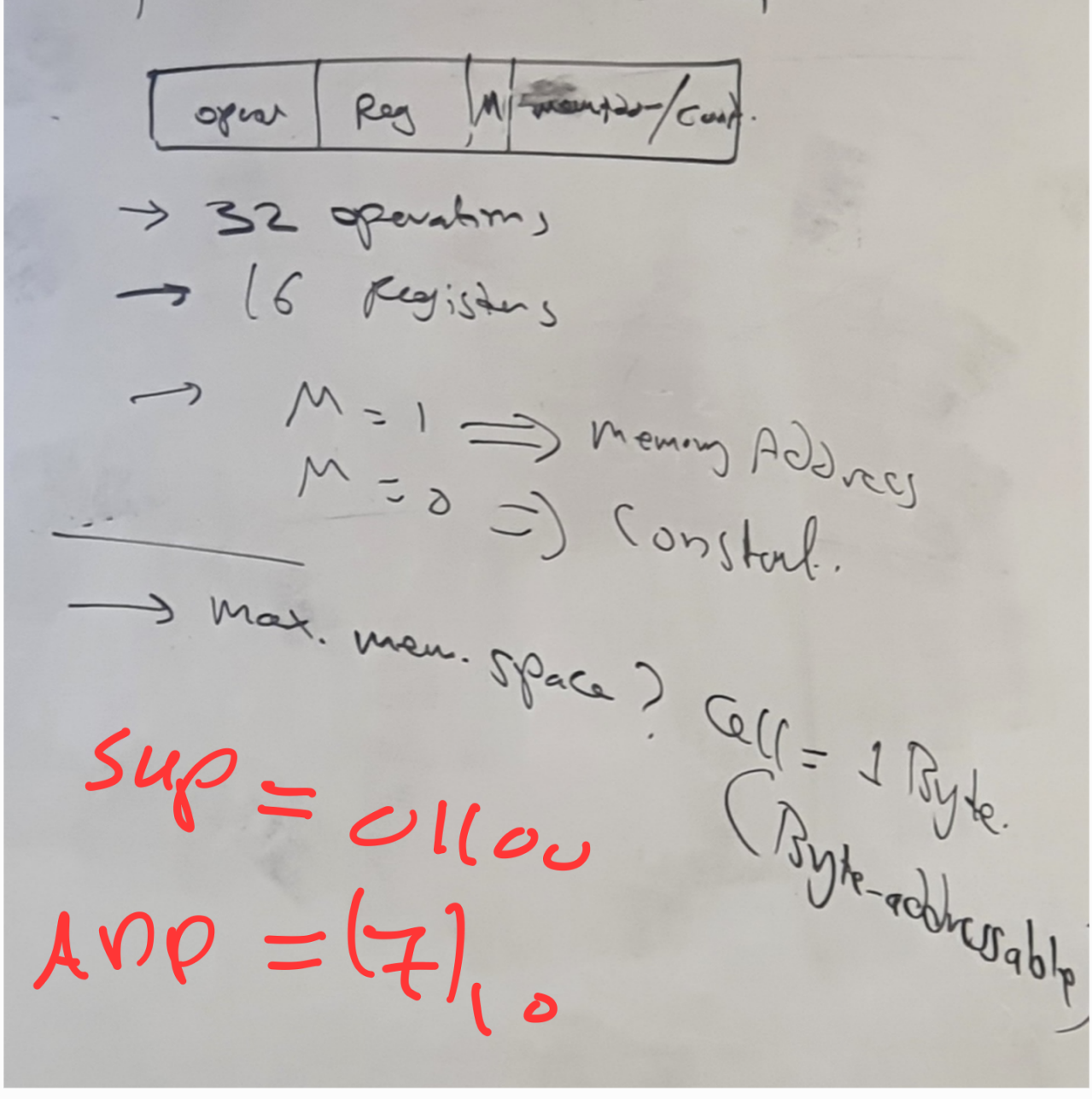
Mode bit

بكره ممكن نعمل لها

operand

Exp:





32 op ⇒ 2^5

⇒ 5 bit

---

16 Reg ⇒ 2^4

⇒ 4 bit

1 bit for Mode

22 bit for last

$\Rightarrow 2^{22}$ cell

$\Rightarrow 2^{22}$ Byte since 1 cell = Byte

$\Rightarrow$ 4 MB

Range constant:

unsigned: $0 \rightarrow 2^{22} - 1$

signed: $-2^{n} \rightarrow 2^{n}-1$

---

write machine code if ADD is $(7)_{10}$

ADD $R_3, [64]$

0011 0011 1 $(64)_{10}$

---

write in assymbly

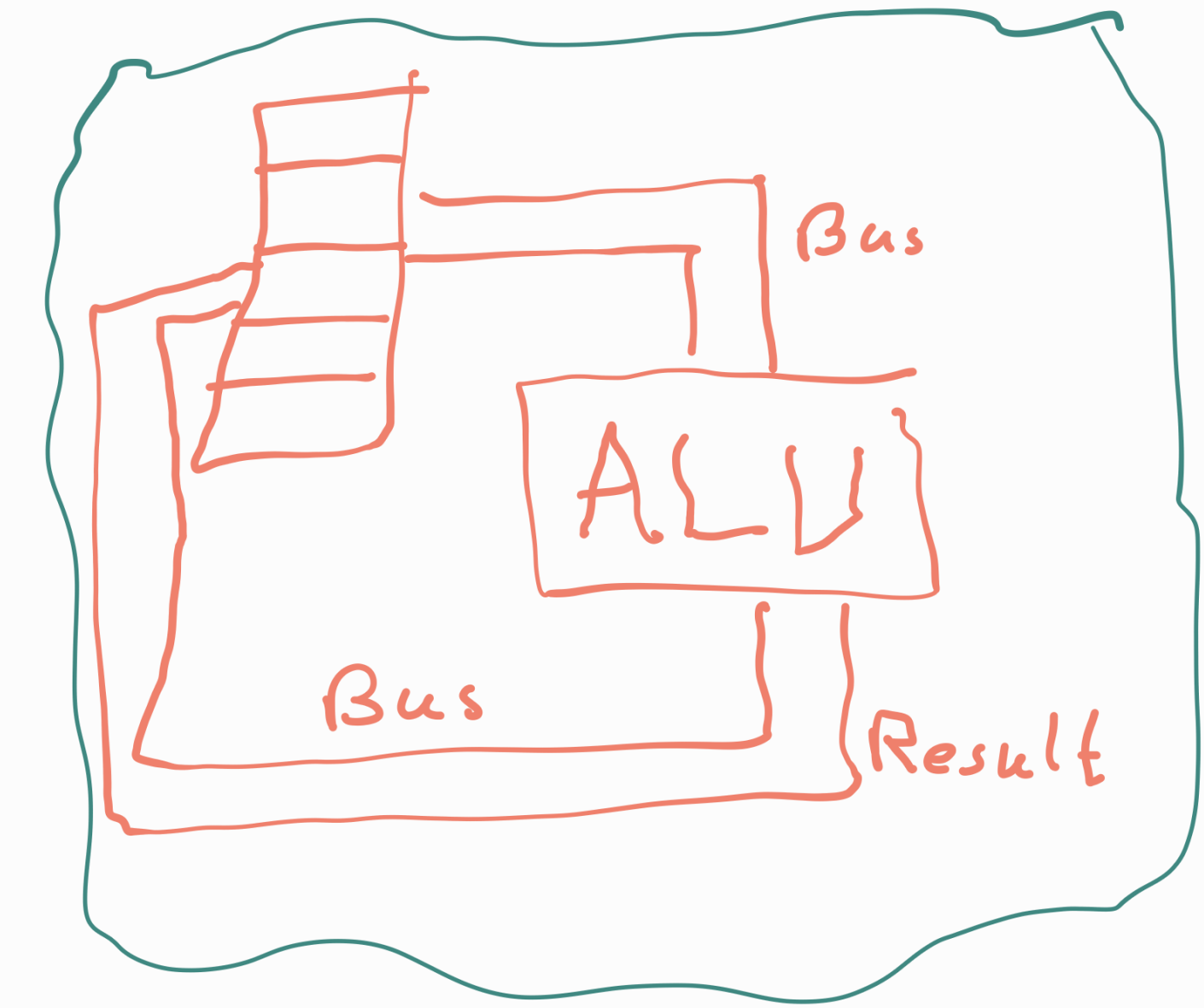
$[67000020]_{16}$

بجھو کیا لکھا ہے

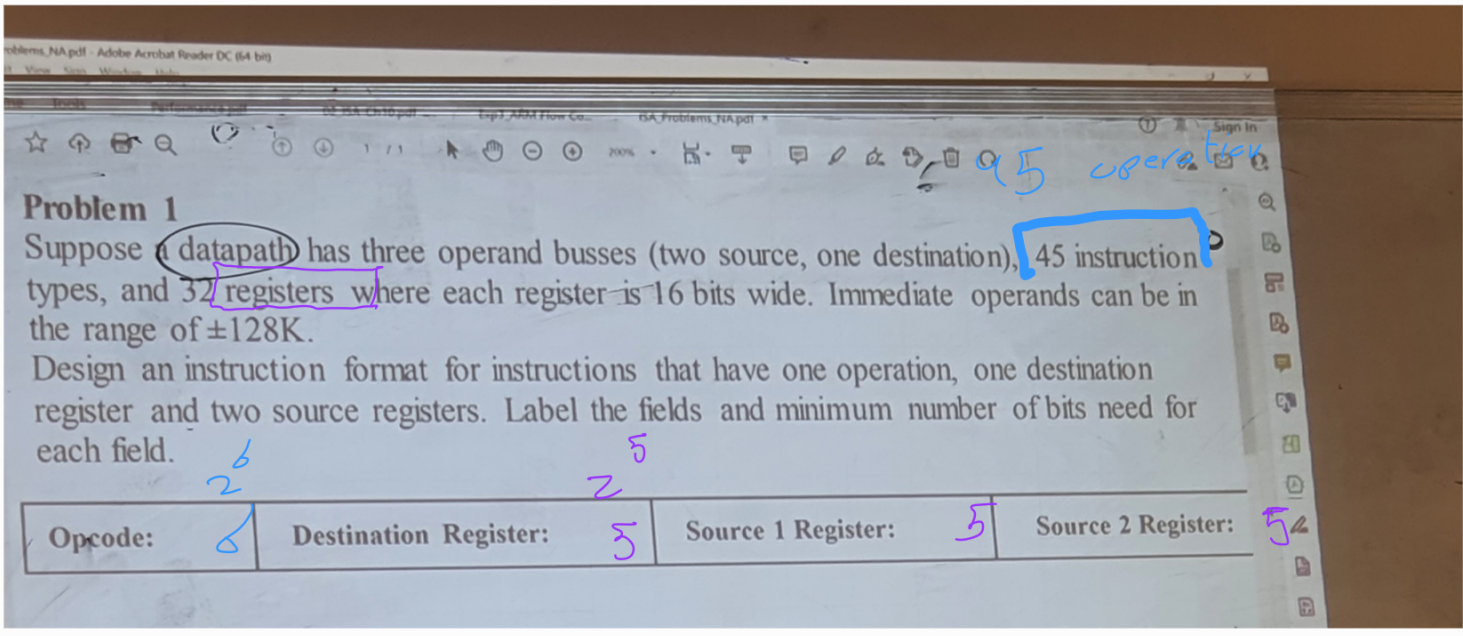
ھاف غلط

01100 1110 0 00000000000000100000

sup R_{14} 32



Datapath

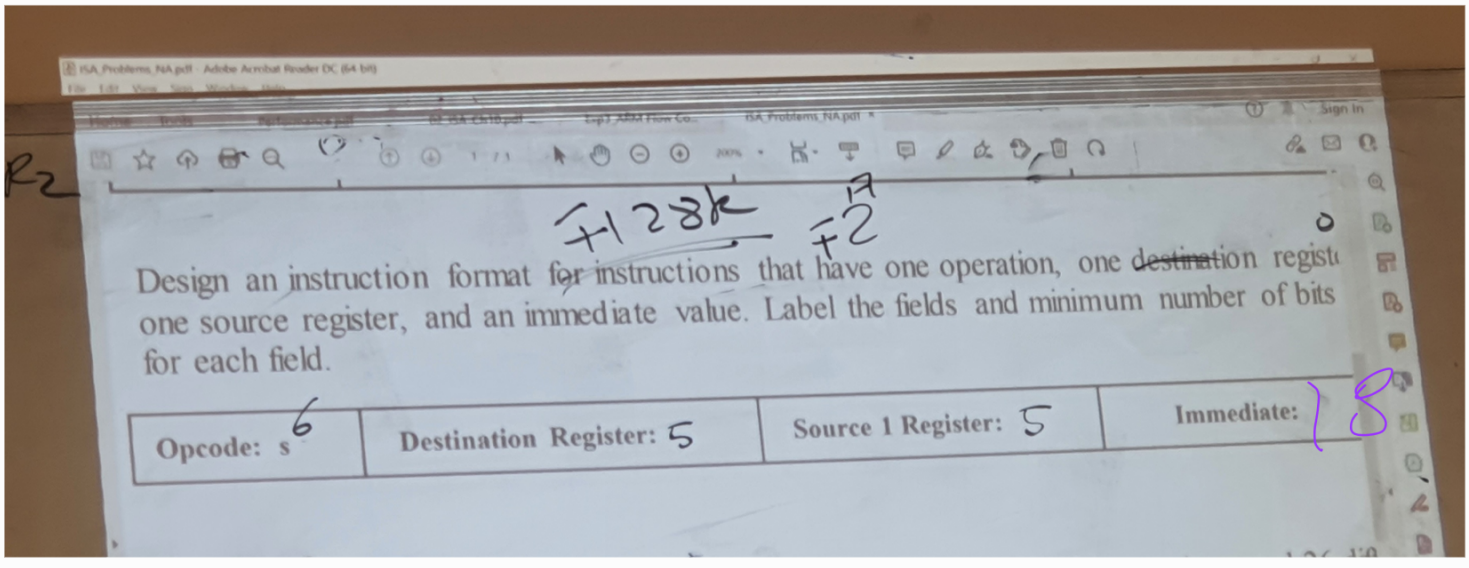


Problem 1

Suppose a datapath has three operand busses (two source, one destination), 45 instruction types, and 32 registers where each register is 16 bits wide. Immediate operands can be in the range of ±128K.

Design an instruction format for instructions that have one operation, one destination register and two source registers. Label the fields and minimum number of bits need for each field.

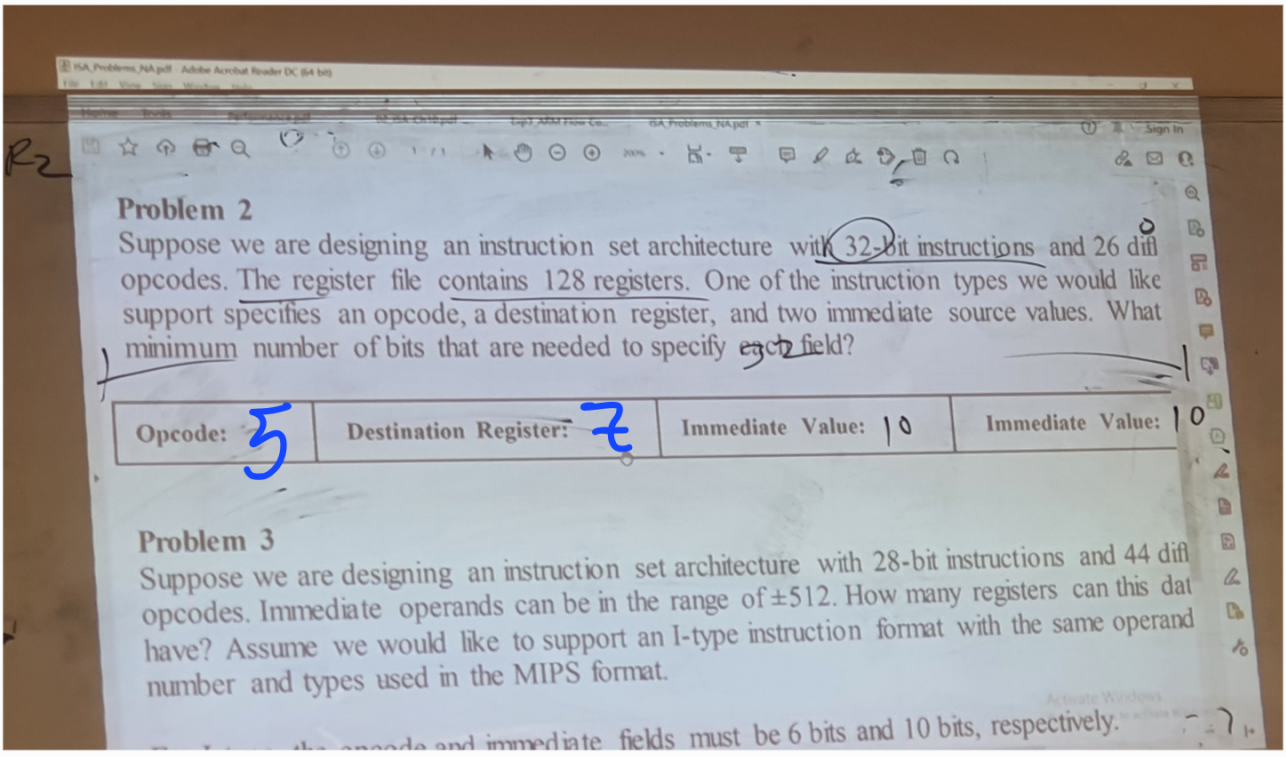
Opcode:	Destination Register:	Source 1 Register:	Source 2 Register:



Design an instruction format for instructions that have one operation, one destination register, one source register, and an immediate value. Label the fields and minimum number of bits for each field.

Opcode:	Destination Register: 5	Source 1 Register: 5	Immediate:

$2^{17} \Rightarrow 18$ bit



Problem 2

Suppose we are designing an instruction set architecture with 32-bit instructions and 26 diff opcodes. The register file contains 128 registers. One of the instruction types we would like support specifies an opcode, a destination register, and two immediate source values. What minimum number of bits that are needed to specify ~~each~~ field?

Opcode: 5	Destination Register: 7	Immediate Value: 10	Immediate Value: 10

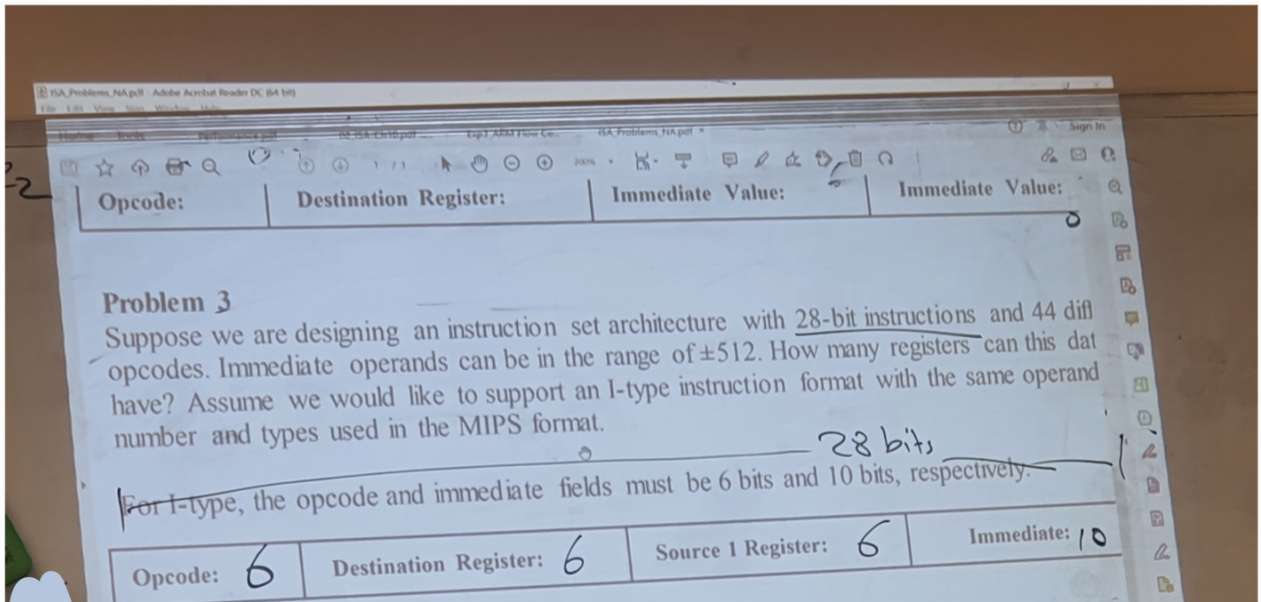
Problem 3

Suppose we are designing an instruction set architecture with 28-bit instructions and 44 diff opcodes. Immediate operands can be in the range of ±512. How many registers can this dat have? Assume we would like to support an I-type instruction format with the same operand number and types used in the MIPS format.

~~For I-type~~, the opcode and immediate fields must be 6 bits and 10 bits, respectively.

if signed

± 512



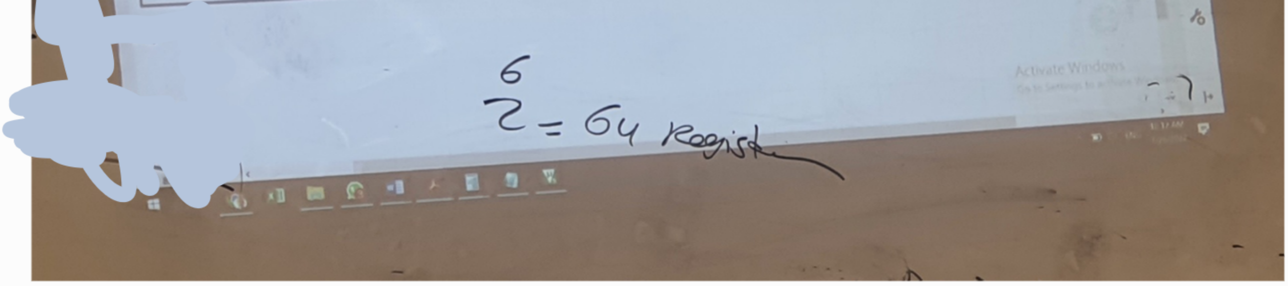
Opcode:	Destination Register:	Immediate Value:	Immediate Value:

Problem 3

Suppose we are designing an instruction set architecture with 28-bit instructions and 44 diff opcodes. Immediate operands can be in the range of ±512. How many registers can this dat have? Assume we would like to support an I-type instruction format with the same operand number and types used in the MIPS format.

~~For I-type~~, the opcode and immediate fields must be 6 bits and 10 bits, respectively. 28 bits

Opcode: 6	Destination Register: 6	Source 1 Register: 6	Immediate: 10

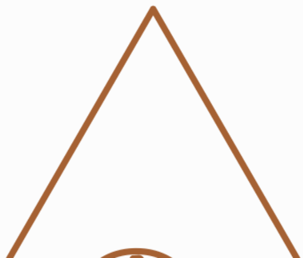


instructor Reg

لتكون فال CPU

بتخزن موقع

لتنفذ



where is instructions and operand?

memory

CPU Registers

I/O

memory

H.D وكيس وغيرها

main memory

نخلي ذاكرة كاش cach

بنقدَر نحط MB لكل
واحد

أكبر إشي

يعتبر بطيء دلالة

بكرنا نستقبل اليوزا

كركة ا 6Hz في 5GHz

مثلاً يحصل وقت

انه تكبيس الكبسة

لسوي فيها ملايين

الحسابات وهو

بحط الانيوت فالرام

Typical Operations

Data Movement	Load (from memory) memory-to-memory move input (from I/O device) push, pop (to/from stack)	Store (to memory) register-to-register move output (to I/O device)
Arithmetic	Data Types: (signed & unsigned) Integer (binary + decimal) (signed & unsigned) Floating Point Numbers Operations: Add, Subtract, Multiply, Divide	
Logical	Not, and, or, set, clear	
Shift	Arithmetic (& Logical) shift (left/right), rotate (left/right)	
Control (Jump/Branch)	unconditional, conditional	
Subroutine Linkage	call, return	
Interrupt	trap, return	
Synchronisation	test & set (atomic r-m-w)	
String	search, compare, translate	

mov store load

if → Control (Jump/Branch)

jump

نفس فكرة الـ call و return بعدين ارجع

02_ISA_Ch10.pdf × Exp3_ARM Flow Co... ISA_Problems_NA.pdf

12 / 37 77.6%

Types of Operand

- Addresses — unsigned int
- Numbers
  - —Integer/floating point
- Characters
  - —ASCII etc.
- Logical Data
  - —Bits or flags

احنا بنحكي cill عبارة عن

c, 1 وهسا احنا شو بدنا

نحط فيهم (Assymply)

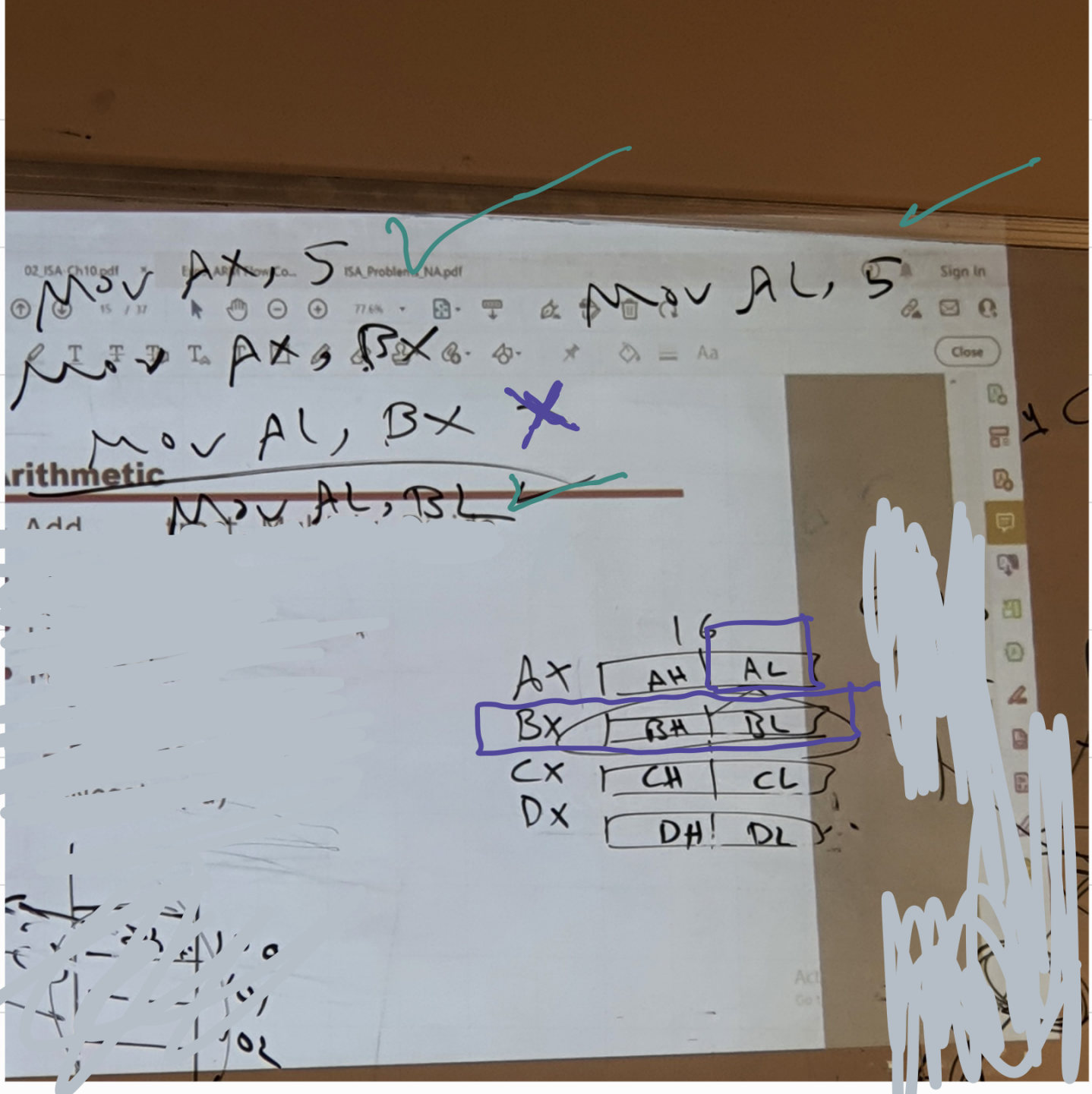
اما فال High level

↓

C, java ....

لما نعرف بحدد مثلاً:

int i و char c

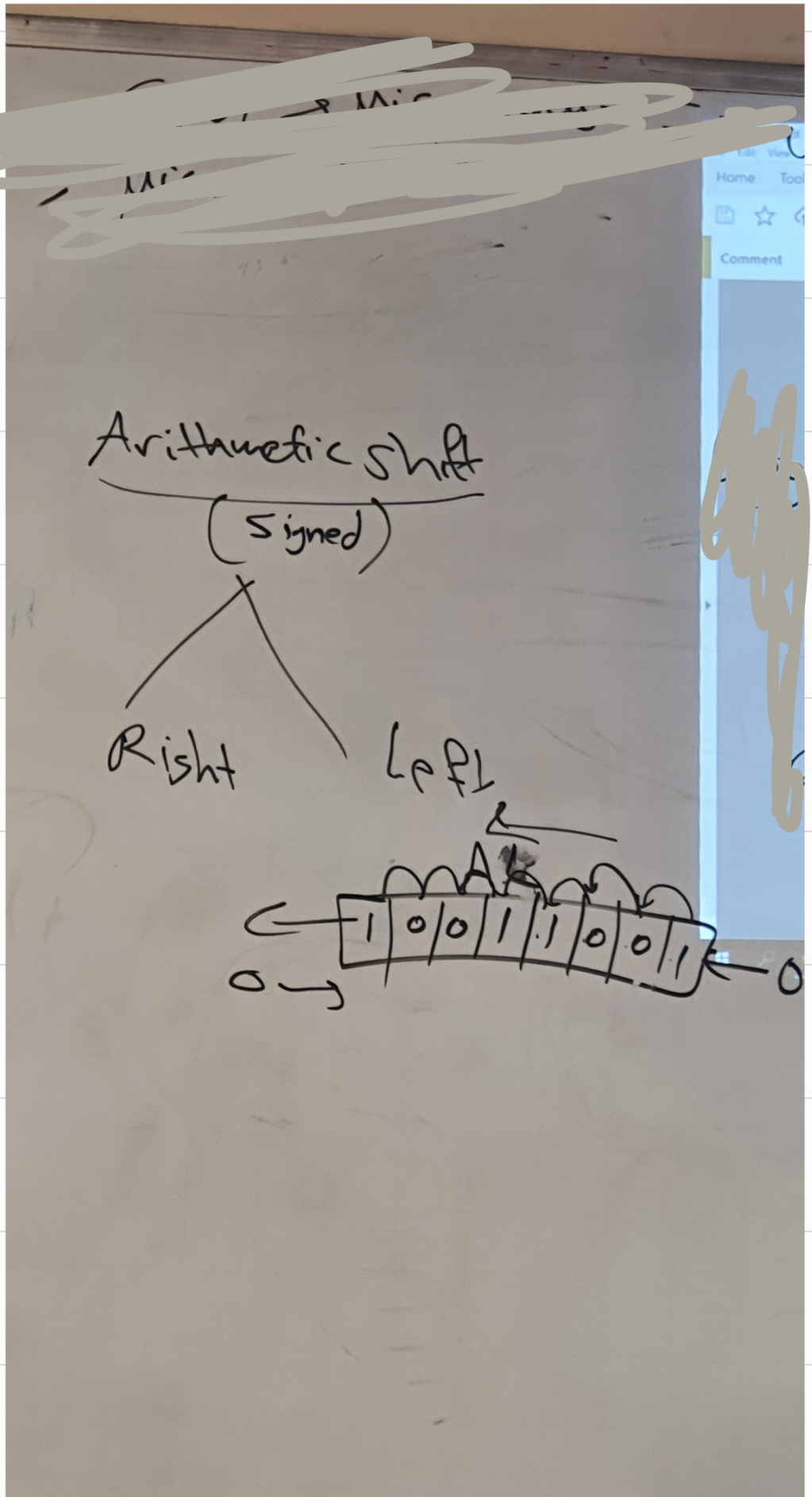


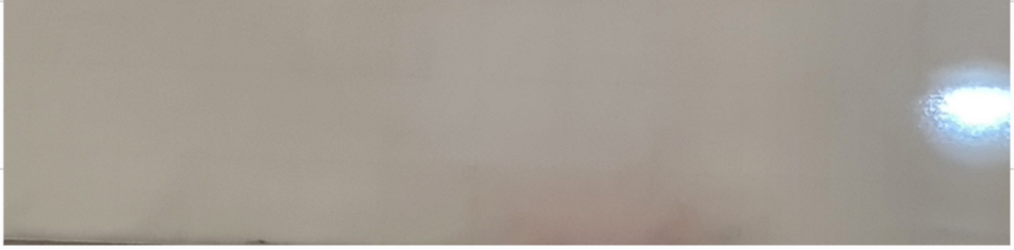
8 bit يعني AL او

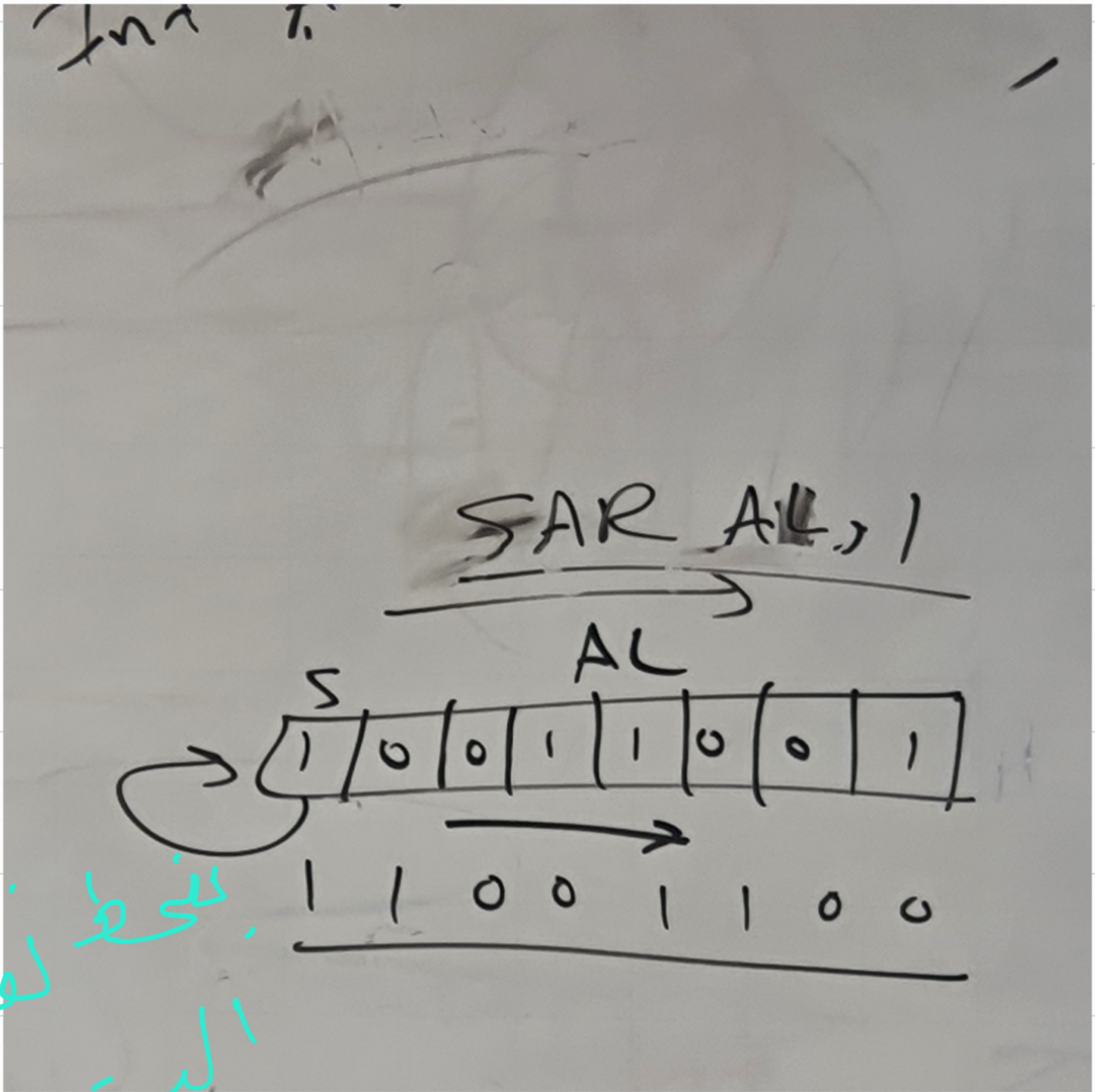
16 bit BX او

NOT Ax

NEG Ax 2's comp.







بنحط نفس البت كله ع الفاضي ع أول واحد يلي هو

sign

shift right

1000

0100

0010

number

2^{n}

عدد الخطوات

shift Lift Right

number $* 2^{n}$

shift left

لما نعمل

اذا تخطى البت يمين كاليسار

يعني overflow

shift Right

= اذا تخطى البت فهاض

بيكون قسمة اعداد صحيحة

Aritmatic ⇒ signed
Logical ⇒ unsigned
} shift

---

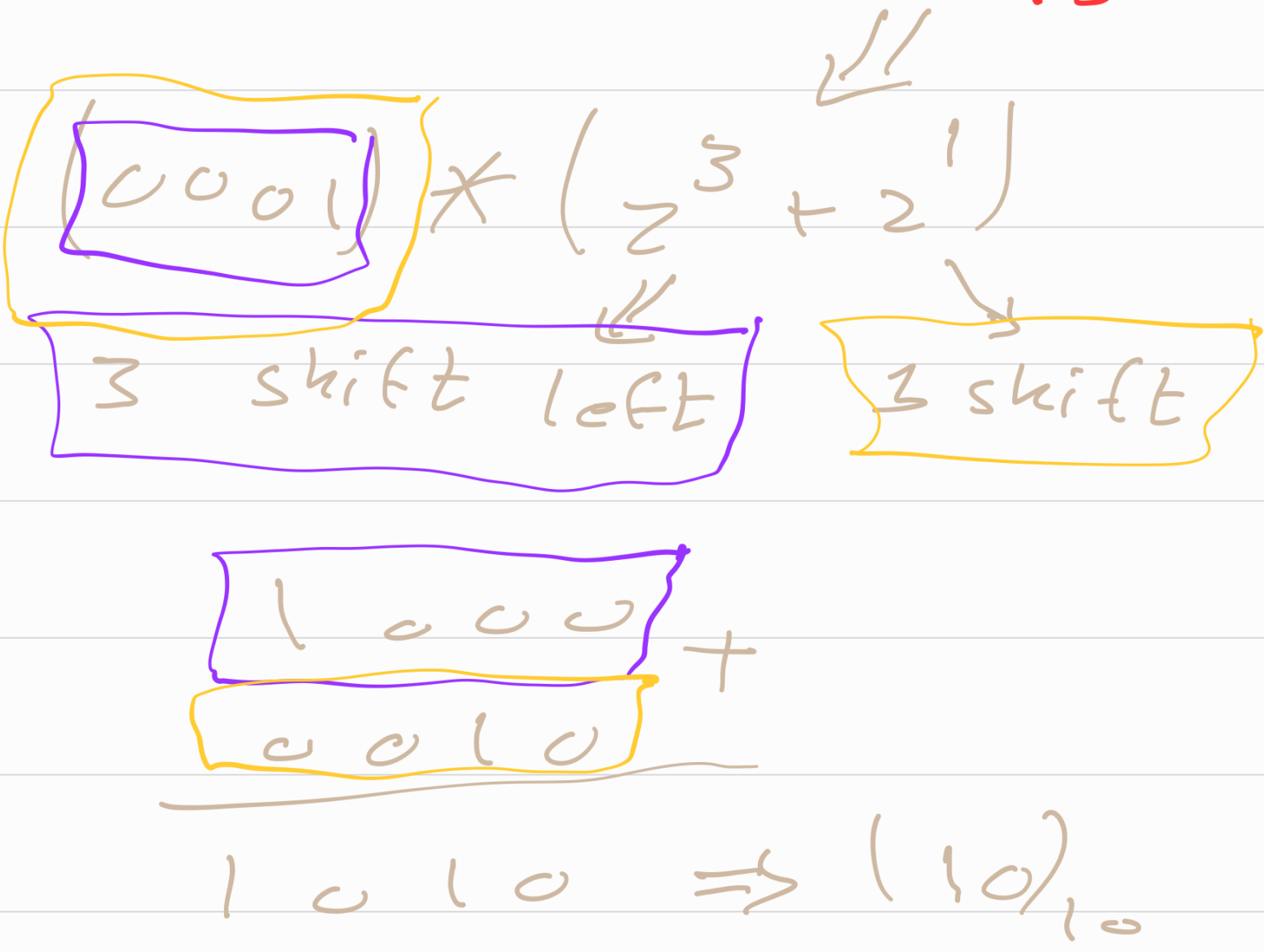
shift Right لما نعمل
ده اذا كان الرقم عايزين نطير

5 ⇒ 0 1 0 1
shift ⇒ 0 0 1 0

5/2
2

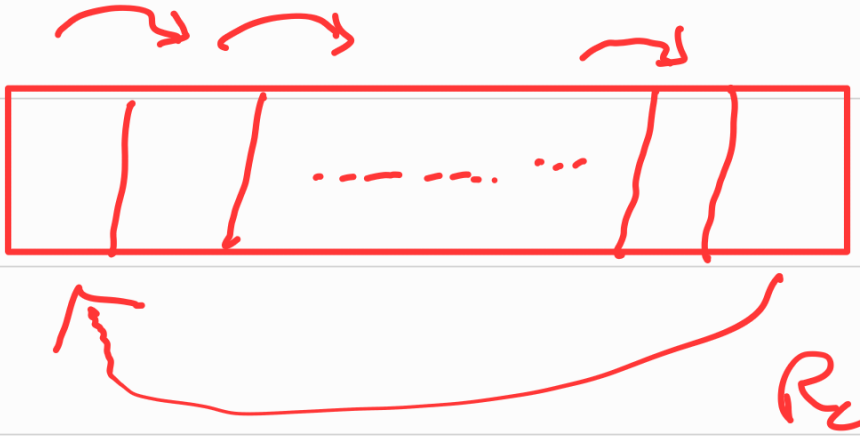
لو بدنا نضرب في 10 بنحولها
لمجموع ... $2^m + 2^n$

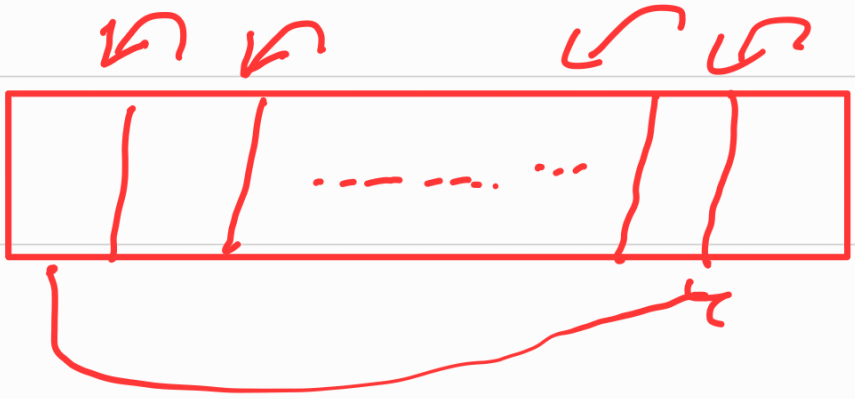
Exp: 0 0 0 1 * (10)



Rotate

كل بت بطلع يرجع عالاول





Rotate left

flags

-ve B (+) Add -ve A

CU

ALU

carry flag

out

+ ve

zero flag (Nor)

sign flag

-ve يعني 1 إذا

+ve يعني 0

overflow flag (Exp)

جمع وطرح العدد نفس الطريقة
والناتج الطريقة بتختلف

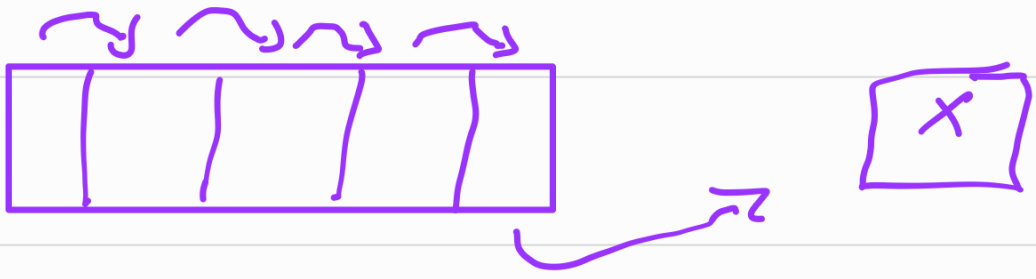
بنقدر نستخدمهم عشان
نفحص العمليات اذا صح
او لا

Exp: لما نجمع وطرح
اذ carry = 1 بجي في علم

قال shift البت الطالعة
بتروح على carry flag

Exp. check odd, even

shift Right ⇒ check carry



طريقة بعيدة عن طرح

And 0 0 0 0 1

0 0 0 0 x

اذا كان even ⇐ 1

odd ⇐ 0

---

clear the register


```
mov  AL, 0
And  AL, 0
Xor  AL, AL
ShL  AL, 8
```

---

clear digit


```
And  1 1 1 1 1 1 1 0
```

And with 1 for all

expext diget to clear

⇒ 0

set to one

OR 1 1 1 1 1 1 1 0

complement

Xor 0 0 0 0 0 0 0 1

JO xyz overflow إذا X yz عددين

JS xyz sign

JZ xyz zero

JC xyz carry

ممكن يكون في n حالة

وكيف

Ex: if equal

```
sub Al, 5
BRZ ——
```

80 JBR loo

81 ———

82 ———

83 ———

loo mov Ax, 1

———

write if (AX > 5)

in assimply

if unsigned => me look at

carry

else

we look at sign

للتسهيل في الشكل جاهزة

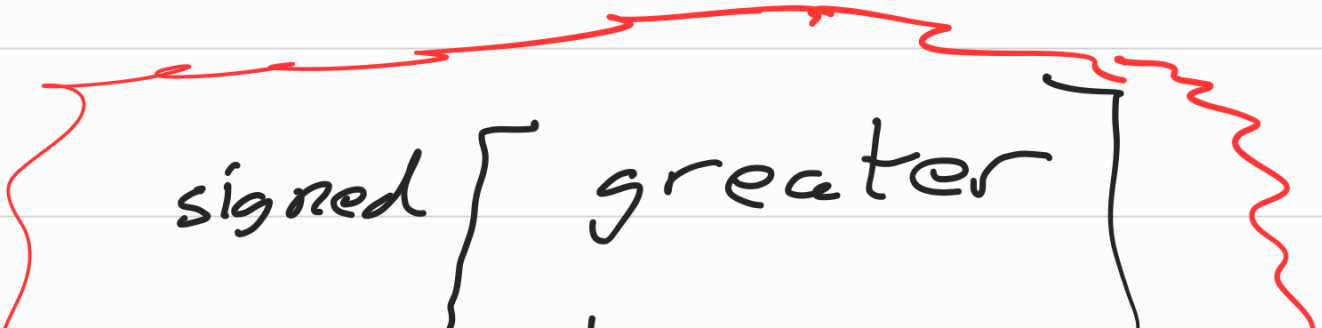
J G ⇐ اكبر signed

J GE ⇐ اكبر او يساوي

J L E ⇐ اصغر او يساوي

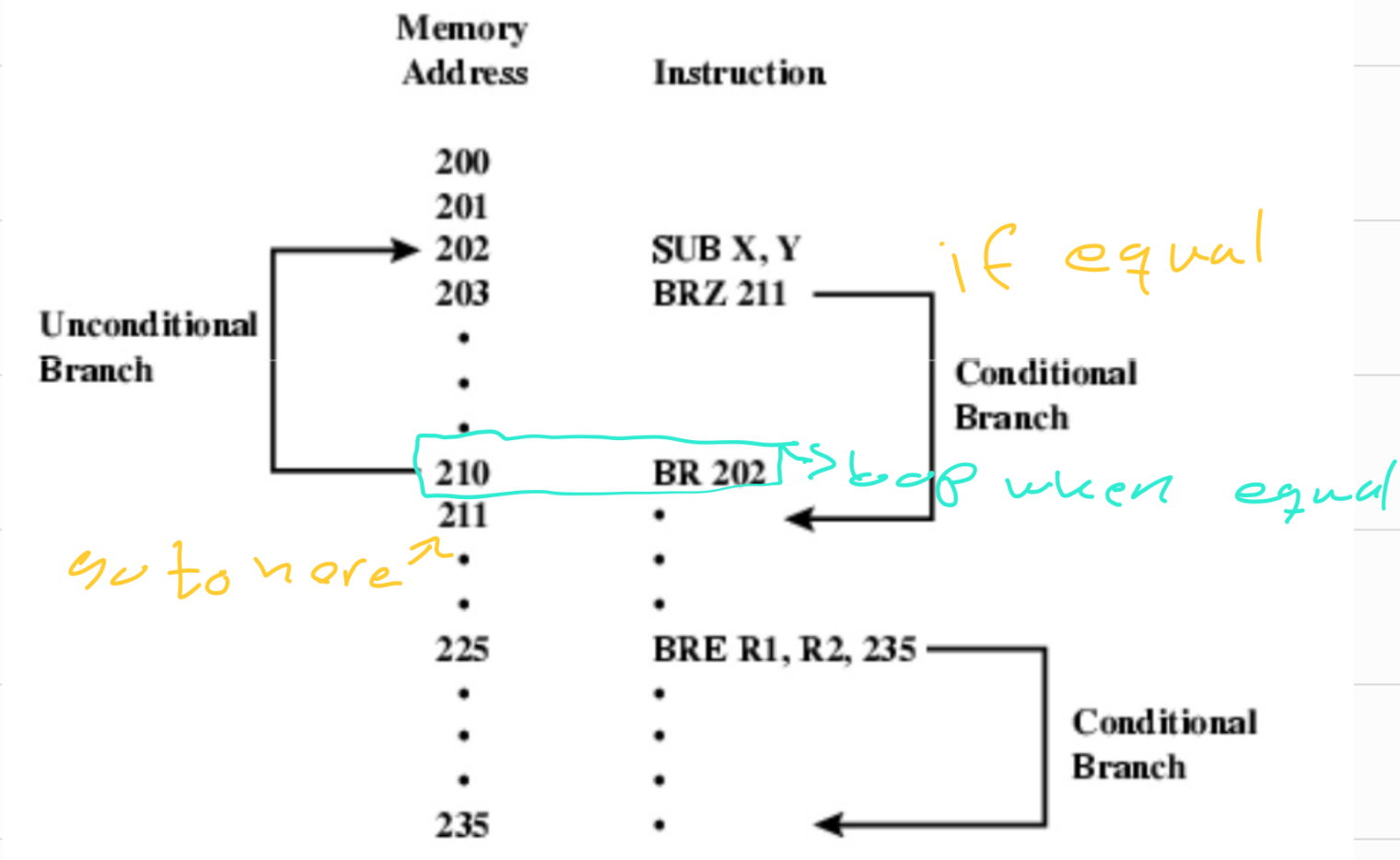
J a ⇐ اكبر unsigned

J b ⇐ اصغر



less

unsigned [below above]



Transfer of Control

- Branch
  - —e.g. **BRZ X** branch to x if result of (ADD,SUB,...) is zero
  - —See next slide
- Skip
  - —e.g. increment and skip if zero ISZ

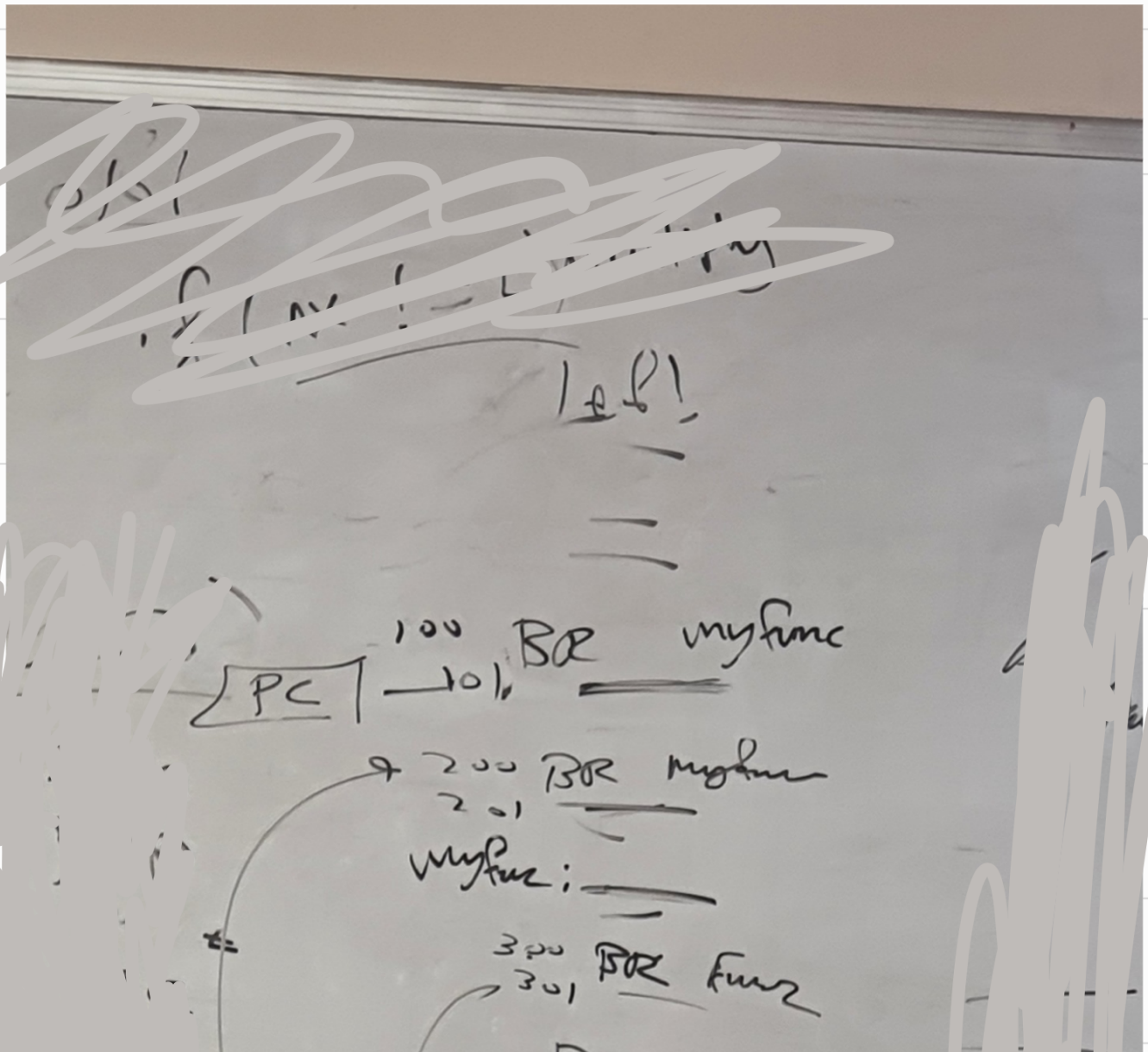
```
301
 .
 .
309     ISZ  R1
310     BR   301
311
```

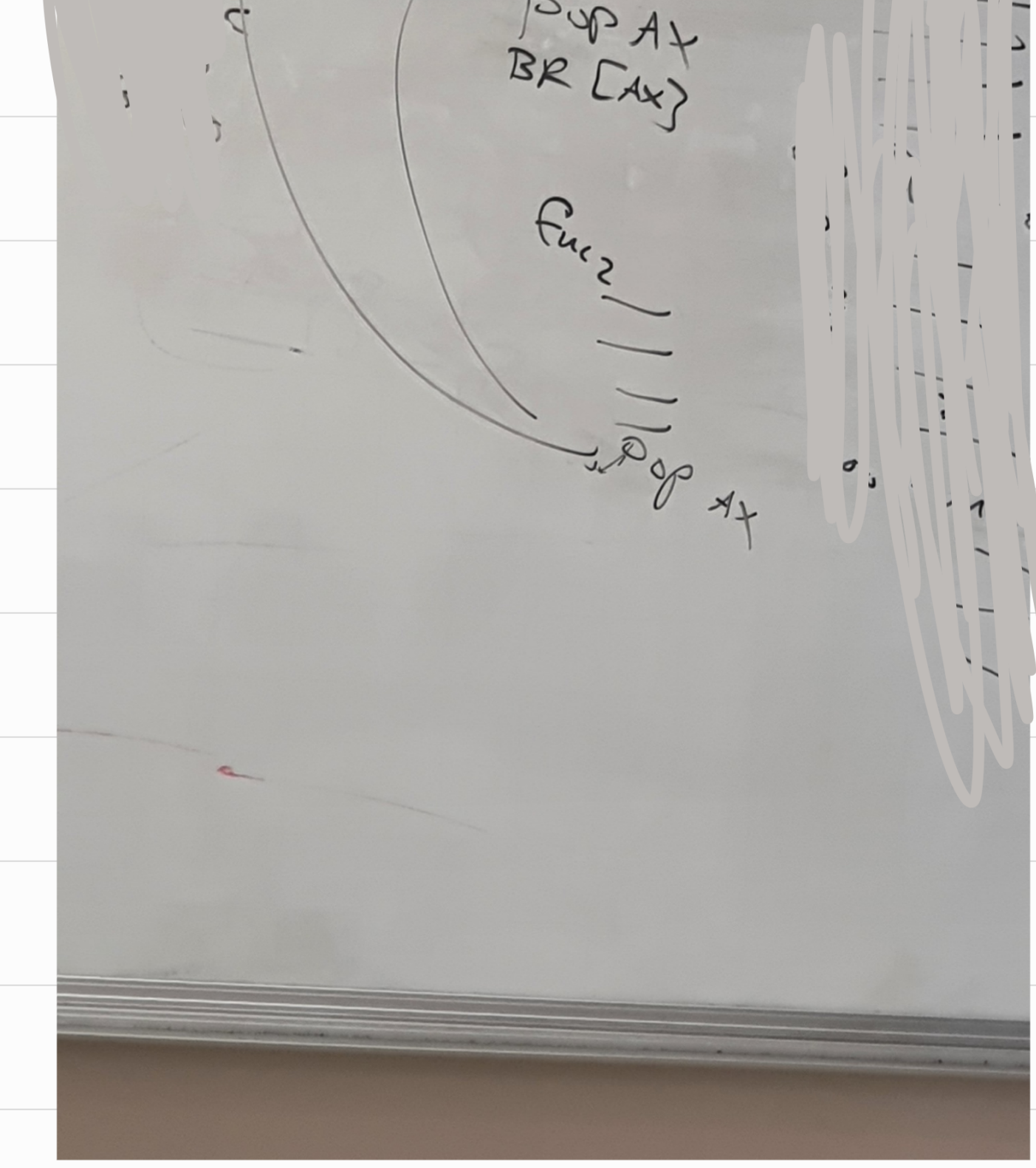
  - * eg. R1is set to -1000, the loop will be executed 1000 times
- Subroutine call
  - —c.f. interrupt call

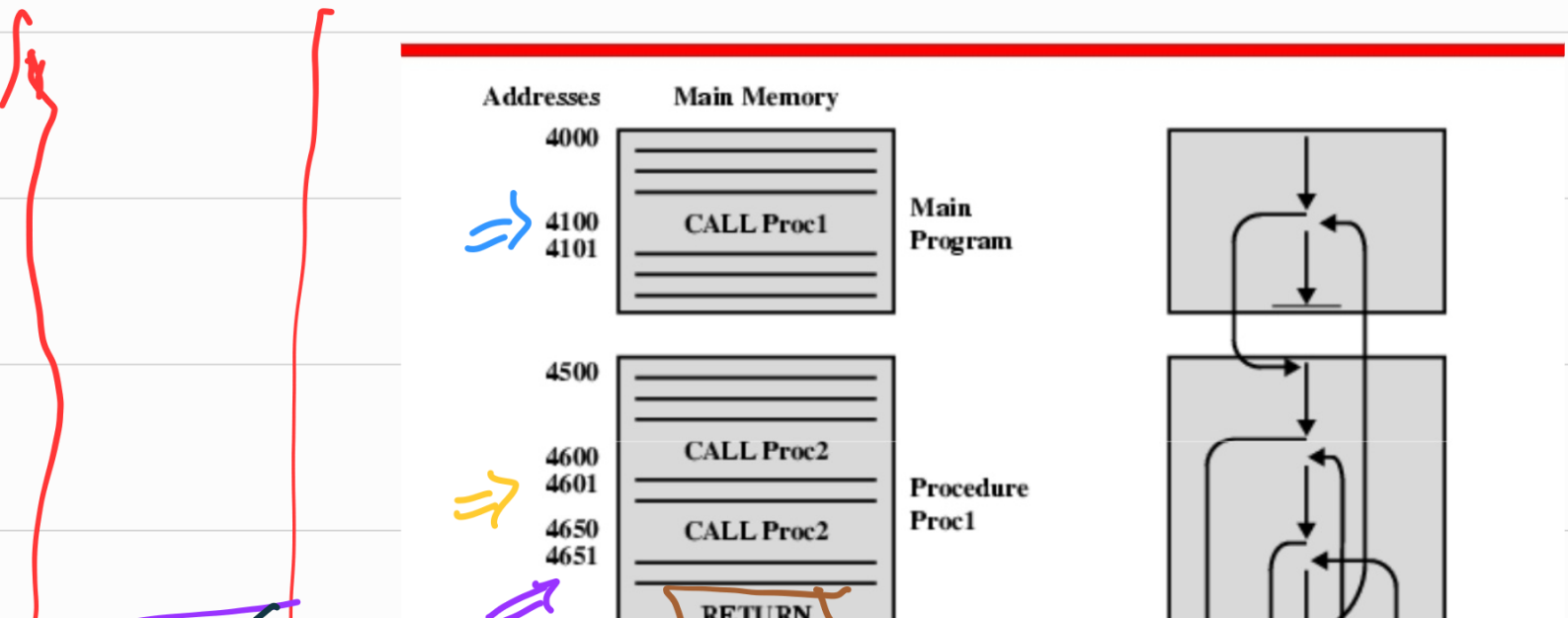
في جزء من الصورة جوز
stack

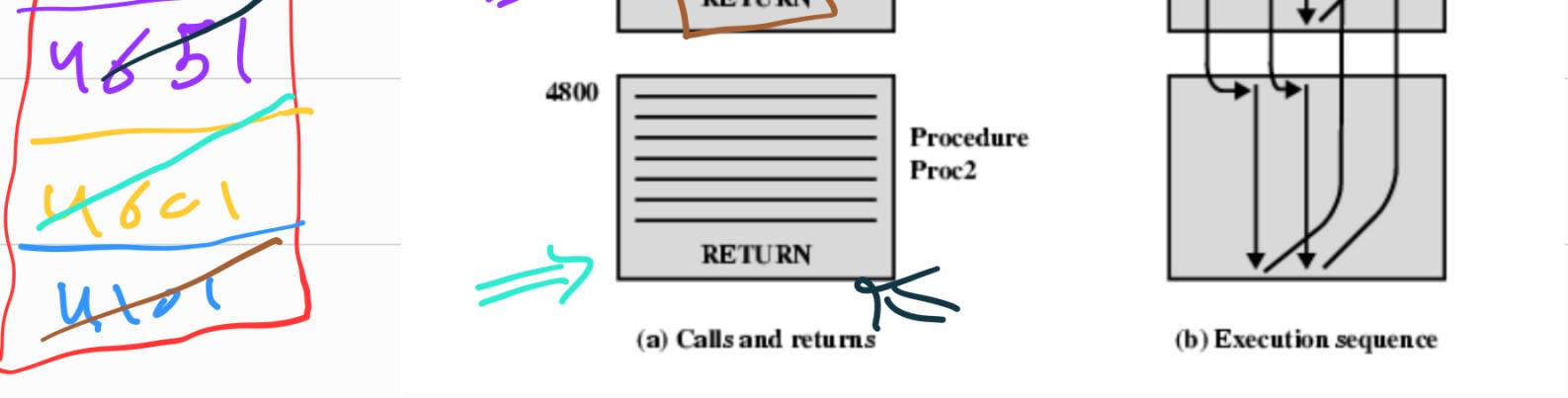
---

there are register called PC
program counter
have address of current
instruction


100 BR myfunc
PC
200 BR myfunc
myfunc:
300 BR func


BR [AX]
Func2
pop AX


Addresses
Main Memory
4000
4100
4101
CALL Proc1
Main Program
4500
4600
4601
CALL Proc2
4650
4651
CALL Proc2
Procedure Proc1
RETURN



in intel:

mul cL

mul BL

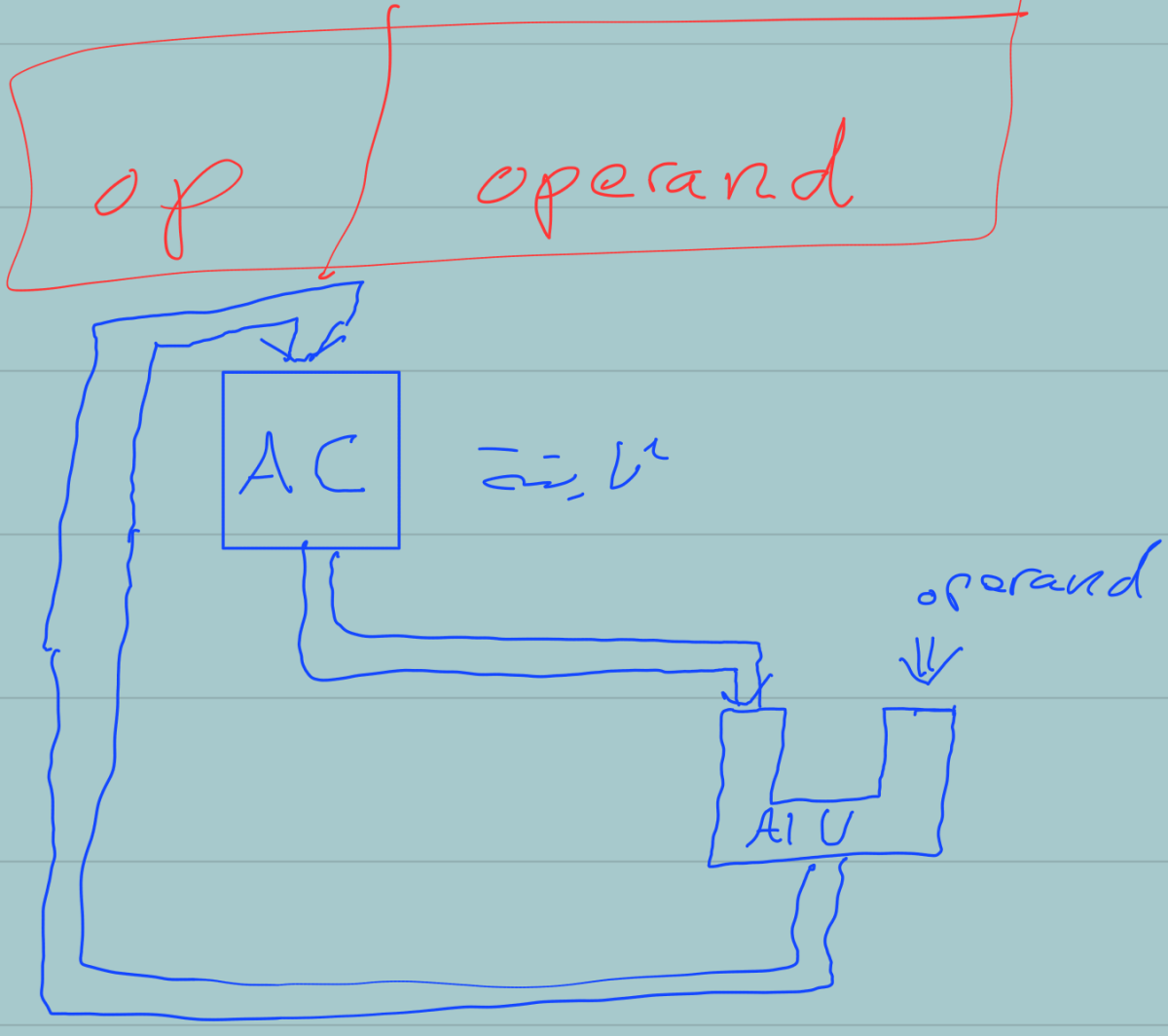
AL => دايما الضرب بتم

=> اذ بنا نضرب رقمين

بنحط واحد في AL

=> و operators بتوخذ اعداد مختلفة من operands

لما نقلل الـ operand بتقل المسرة لانه بيكون متعود على عدد معين فالكود بيكبر



---

operator

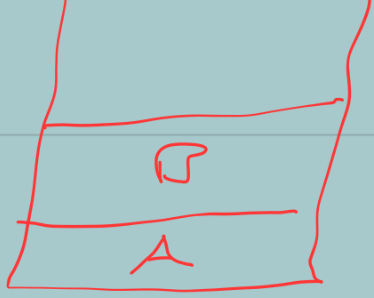
Stack machine

- كل اشي بيتخزن ف stack

فهو بيعبي بيه كل اشي

وبنزلها

Exp ADD



⇐ بطلع (pop) A و B وبجمعهم push

very important

Top ⟵ Top − 1 of Top

Exp: $Y = \dfrac{A - B * C}{D + E}$

where Y, A, B, C, D, E memory

By 3 Address, 2, 1, 0

3 Address

Mul Y, B, C

Sub Y, A, Y

ADD T, D, E

Div Y, Y, T

---

2 Address

Mul B, C

sub A, B

ADD D, E

Div A, D

Mov Y, A

1 - Address (Accumulator

load B نخزن B في Ac

mul C

store T_1 $T_1 \leftarrow Ac$ نخزن

load A

sub T_1 $Ac = Ac - T_1$

store T_1

load D

ADD E

Store T_2

load T_1

Div T_2

store y

---

0 - address

push A

push B

push C

Mul

Sub

push D

push E

Add

Div

pop y

⇒ post - fix

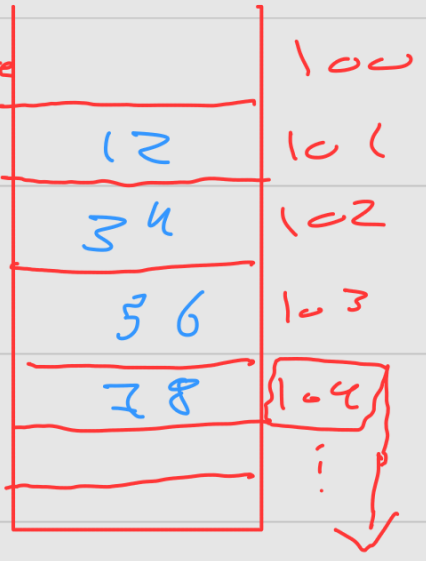
$(1\,2\,3\,4\,5\,6\,7\,8)_{16}$

$R_0 \rightarrow 32$ bit

12	34	56	78

Big endian

cell => 1 Byte



Mov [100], R_0

ثم تخزين أولا

Endian

→ big - endian

least sig. byte ⇒ high Address

(الأكبر)

→ Little - endian

intel use it

least sig. byte ⇒ lowest Address

$R_0 \rightarrow 32$ bit

12	34	56	78

cell => 1 Byte

little endian

Mov [1003], R0

34	1003
12	1004
	⋮

Address

we do that just when data larger than cell size