CURS 1. → 5 oct. 2022

B_2 → reprezentarea calculatorului

B_{10} → interpretarea B_2

Tip dată ↳ o structură (și un domeniu de valori)
↳ + o operație asociativă

programare pe obiect: moștenire polimorfism.

EU = executive unite

tot ajunge în ALU ⇒ aritmetică (op. simple)

Bit → unitate primară de reprezentare a informației

* 1024 *

Octet → cea mai mică unitate accesibilă la niv. memoriei

* memoria calculatorului este organizată pe octeți.

Registrii → capacități de memorare de la niv. procesorului foarte mici ca dimensiune de memorare (8, 16, 32, 64 biți) însă f. rapide ca viteză de acces la info.

→ au rolul de a stoca temporar operanzii cu care lucrează în mod curent un procesor (date, coduri de comenzi, adrese)

CURS 2 → 12. oct. 2022

Registrii generali ai EU

RAM → Random Access Memory.

Viteza de acces / timpul la orice zonă de memorie este același, indiferent de poziția față de începutul memoriei

→ suportă READ & WRITE într-o ord. aleatoare

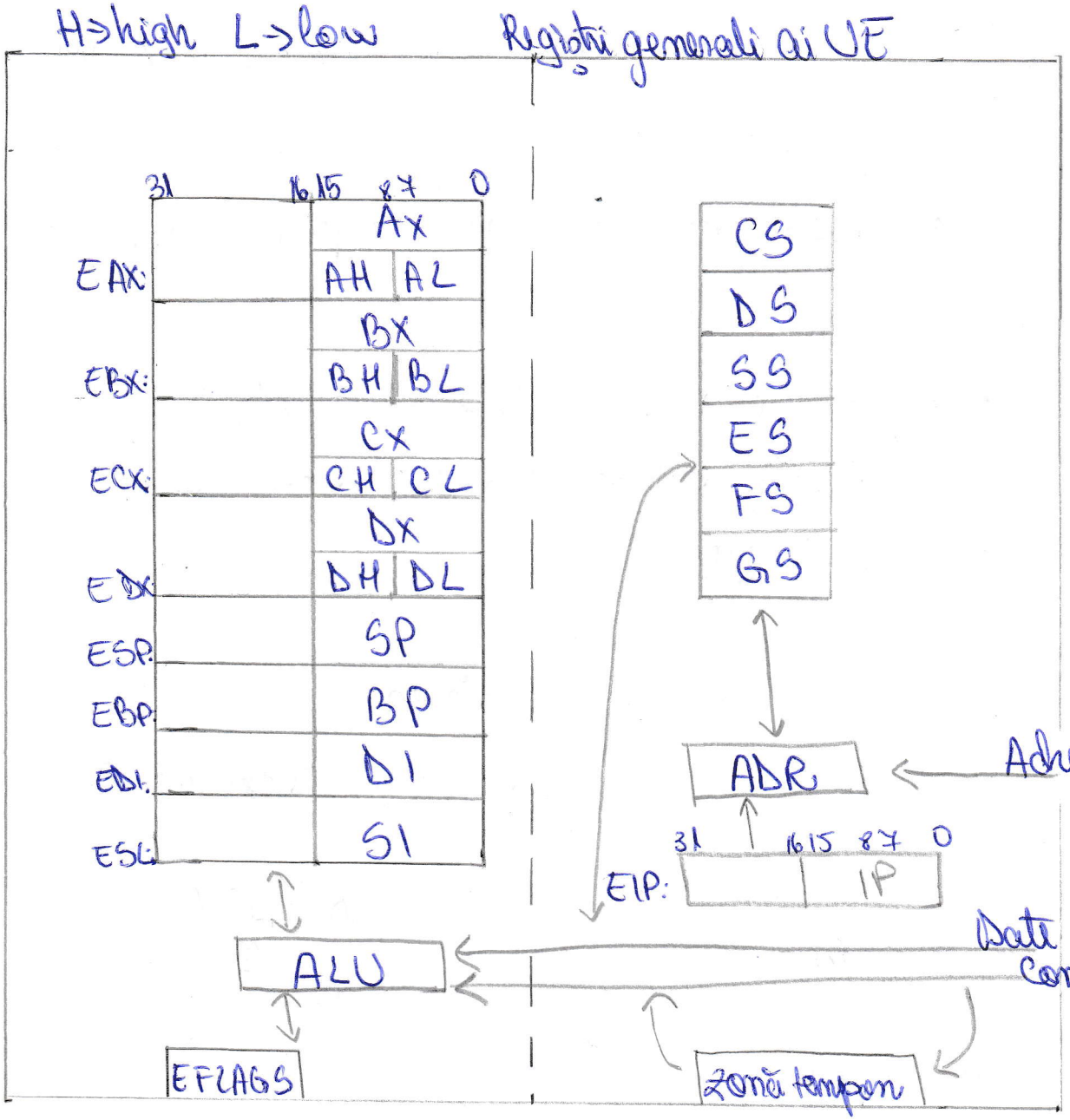
- EU & BIU lucrează simultan
- memorie cash → depozit care face legătura, aduce mai mult pt "rezervă".
  → cash al memoriei video.

Registrii generali
↓
nu sunt limitați

ex: înmulțire → rezultatul este pus în EAX / EX.
↗
extended
- caracteristică a variantei pe 32 biți

Am pornit de la programarea pe 16 biți

H->high L->low



EAX -> registrul acumulator

- se folosește drept unul dintre operanzi de majoritatea instrucțiunilor

EBX -> registrul de bază

- punct de pornire pt. array.
- sistem de adresare în spații

ex: a[39] = * (a+39)

↑ bază (adresă de start)

= Doar C lucrează cu adrese, făcut din necesitate pt UNIX =

ECX → registru contor
- utilizat în instr. repetitive pe post de „variabilă de ciclare"

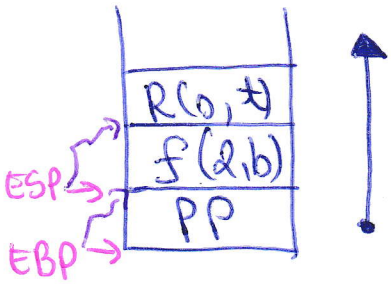
EDX → registru de date (Data Register)
- împreună cu EAX se folosește în calculele ale căror rezultate depășesc un dublu cuvânt.
- „extinde" EAX

ESP, EBP } destinați lucrului cu stiva (delimitează cadrul de stivă).

STIVA → disciplină specială de acces LIFO
↑
last in, first out.

COADA → FIFO.
↑
first in, first out.

funcțiile procedurale ⟷ limbaj de abstractizare



ORDINEA LIFO

! procesoarele se concentrează pe stivă, nu pe coadă

x-ss.

Stack pointer (SP) → puncteaza pe ult. elem. din stiva
Base pointer (BP) → puncteaza spre baza stivei

EDI → destination index
ESI → source index
} registrii de index, utilizati pt. a accesa elem. din siruri de octeti, cuvinte sau dublucuvinte

FLAG → „un registru", un indicator reprezentat pe un bit

EFLAGS → are 32 biti, se utilizeaza doar 9.

31	30	...	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	...	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

Exemplu

```
 1001|0011+
 0111|0011
(1)0000|0110
```

se duce in carry flag (de transport)

UOE - ult. op. efectuata
AF = 0 (auxiliary flag)
PF = 1 (perity flag)
CF = 1 (carry flag)

AF: indica daca exista transport de la bitul 3 la bitul 4
CF → carry flag (bitul de transport)
ZF → zero flag → daca a fost 1 → 0 - false
↳ daca a fost 0 → 1 - true
SF → Sign flag (bitul de semn) → 0 - pozitiv
↳ 1 - negativ

! Pt. procesor 0 este un număr pozitiv

TF → trap flag (flag de depanare, utilizat în depanatoare)
→ 1 = procesorul se oprește după fiecare execută

IF → interrupt flag. → utilizat sub 16 biți

DF → direction flag (de direcție) → 0 = ascendent
→ 1 = descendent

OF → overflow flag (flag de depășire)
→ dacă UOE nu a încăput = 1, altfel = 0.

(*) Reprezentare → B_2
Interpretare → B_{10} = cu / fără semn

! CARRY → fără semn
OF → cu semn

CF, PF, AF, ZF, OF → arată ce s-a întâmplat

DF → ce se va întâmpla
↳ { CLD → 0
STD → 1

IF
↳ { CLI → 0
STI → 1

Carry flag → dublu posibilitate
- îl pot seta
- spune ce s-a întâmplat.