INTREBARI & RASPUNSURI ASC // EXAMEN GRILA

1. **Scrieti lista principalelor registre folosite in programul ASM (notatie, denumire)**

Principalele registre folosite in programul ASM sunt:
ax - numit si acumulator
bx - numit si base
dx - numit si data
sp - pointer catre varful stiveri
ip - pointer catre urmatoare instructiune ce urmeaza a fi executata
ds - retine adresa de inceput a segmentului de cod .data
ss - retine adresa de inceput a stivei

2. **Ce reprezinta al si ah?**

AL reprezinta Accumulator Low si AH reprezinta Accumulator High fiecare avand 8 Low byte si high byte

3. **Ce puteti spune despre variabila SS3 , definita prin SS3 db 18 dup(?)**

SS3 este un sir 18 bytes neinitializat (db = define BYTE, 18 = nr de elemente, ? - neinitializare)

4. **Care este registrul implicit folosit in instructiunile uzuale din ASM?**

Acumulatorul - AX

5. **Cati biti are un cuvant format din 32 Bytes?**

256

6. **Cati biti de adresare sunt necesari pentru a accesa 64kb de memorie?**

16

7. **Cati bytes sunt intr-un cuvant dintr-un sistem cu memoria organizata pe 64 biti ?**

8

8. **Care este diferenta dintre directive si instructiuni in ASM si cum se scriu in fisierul sursa ?**

O directivă nu se traduce printr-un cod executabil şi în consecinţă NU se execută de către procesor.
Instructiunile se traduc prin cod si fac parte din limbajul ASM
Directivele vor incepe cu . in sursa

9. **Care sunt registrii ASM si care este utilitatea fiecaruia ?**

ax - numit si accumulator(la instructiuni aritmetice,de exemplu,la inmultire si adunare)
bx - numit si base(folosit in adresarea indexata)
cx - numit si count(pastreaza 'bucla' de numarare(count) in operatiuni iterative)
dx - numit si data(folosit impreuna cu registrul ax acumulator la operatiile de inmultire si impartire ce implica valori mari)

10. **Care este secventa de instructiuni ASM ce inlocuieste „if" din C ?**

Secventa ce inlocuieste 'if' din C in ASM este 'CMP' + JMP conditional

11. Cum se numesc expresiile dintr-un program sursa ASM care incep cu un punct si cum se numesc cele fara punct?

```
.data
…
Add al,5
```

Expresiile dintr-un program sursa ASM care incep cu un punct si cele fara un punct se numesc (puncte directive ; fara instructiuni)

12. Care este registrul implicit folosit intr-o instructiune de adunare (ADD)?

Registrul implicit folosit intr-o instructiune de adunare (ADD) este ax.

13. Care este rezultatul secventei de instructiuni:
Mov al,8
Add al,5 si unde este acesta salvat?

Rezultatul secventei de instructiuni este 13 si este salvat in al.

14. Ce reprezinta dl si dh?

Dh=cel mai semnificativ byte al lui dx(h=most)
Dl=cel mai putin semnificativ byte al lui dx(l=least)

low byte si high byte

15. Ce puteti spune despre variabila VV,definita prin: VV dw 50 dup(?)

VV este un cuvant de 50 de bytes neinitializat(dw=define word,50=nr.elemente,?=neinitializare)

16. Scrieti lista principalelor registre folosite in programul ASM (notatie,denumire si utilizare)

17. Care este registrul implicit folosit de instructiunea de adunare in ASM?

Este acumulatorul(ax)

18. In assembler ,o secventa conditionala corecta cuprinde instructiunile in urmatoarea ordine?

a.) CMP,INC
b.) JMP,CMP
c.) INC,CMP
d.) CMP,JMP
e.) INC,JMP

19. Cate intrari poate avea un demultipexor ?

a.) Una
b.) Niciuna
c.) Oricate
d.) Doua

20. Cate locatii de memorie pot fi selectate cu o magistrala de adrese pe 20 biti ?

256 KB

21. Degrevarea CPU pentru transferul din memorie se face catre :

a.) DNA
b.) ADN
c.) DMA
d.) Bus-ul de date
e.) Bus-ul de adrese

22. Pregatirea unei operatiuni aritmetice in assembler presupune :

a.) Apelarea functiei de control
b.) Mutarea unui operand in accumulator
c.) Mutarea unui operand in memoria cache
d.) Folosirea operatorului vectorial
e.) Mutarea codului operatiunii in registrul de instructiuni

23. Cate iesiri poate avea un multiplexer?

a.) Una
b.) Niciuna
c.) Oricate
d.) Doua

24. Cati bytes are un registru din programul ASM al dvs?

2 Bytes(16 bit)

25. Ce este ".data" si ce este "mov" in assembler. Care sunt caracteristicile lor?

.data - sectiunea de program in fisierul obiect sau in memorie ce contine variabile
mov a, b - muta continutul variabilei b la adresa in memorie a variabilei a

.data - segmentul de date
mov a,b - incarca o valoare intr-un registru sau locatie de memorie

26. Ce puteti spune despre variabila TTz , definita prin: TTz dt 5 dup(?)

dt - define ten
TTz = 5 elemente de cate 10 bytes, neinitializare

27. Care este rezultatul unei operatiuni SI (AND) pe biti dintre A si B.
Adica A&B=
unde:
A=10011010
B=10100110

10000010

28. Scrieti valoarea in hexazecimal

82 = 0x52 (in hexa)

29. **Cate linii de adresa sunt necesare accesarii unei memorii de 32 GB?**

    35

30. **Care sunt mintermenii functiei F din tabelul:**

x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

    X'Y'Z', X'YZ, XYZ

31. **Cate iesiri are un demultiplexer pe 4 adrese?**

    16

    (, depinde ce se intelege prin 4 adrese , poate fi si 4)

32. **Ce sunt si ce inseamna:**

    .stack
    .data
    .code

    Acestea sunt directive.
    .data- sectiunea de program in fisierul obiect sau in memorie ce contine variabilele
    .code- un text ce face referinta la o parte a memoriei sau la un fisier obiect ce contine

    instructiuni de executare
    .stack- o structura de date

33. **Ce inseamna:**

    int 21h

    si cum este pregatita (instructiuni premergatoare)

    int 21h- Principala intrerupere DOS deoarece da acces la toate functiile.(in proiect am scris ca iese din program prin intrerupere fara erori)
    Se incarca in registrul AH valoarea 09H inaintea folosirii

34. **Descrieti (ce inseamna) fiecare parametru al instructiunii:**
    vect5 dw 25 dup(?)
    si spuneti ce rezulta

    vect5-declara un vector cu 5 elemente
    dw- cuvinte de 16 biti
    25 dup(?)- duplicarea de 25 de ori a ce se afla in paranteza
    ?- valoare neinitializata
    => vector de 5 elemente fiecare avand lungimea de 16 biti

35. **Cati biti are un cuvant format din 4 Bytes?**

    32

36. **Cati biti de adresare sunt necesari pentru a accesa 64MB de memorie?**

    26 biti

37. **Cati bytes sunt intr-un cuvant dintr-un sistem cu memoria organizata pe 16 biti ?**

    2

38. **Care este diferenta dintre directive si instructiuni in ASM si cum se scriu in fisierul sursa?**

    O directivă nu se traduce printr-un cod executabil şi în consecinţă NU se execută de către procesor.
    Instructiunile se traduc prin cod si fac parte din limbajul ASM
    Directivele vor incepe cu . in sursa

39. **Care este utilitate implicita a registrului AX?**

    registrul acumulator - operator implicit in unele intructiuni

40. **Ce puteti spune despre variabila X1 ,definita prin X1 db 18 dup(?)**

    X1 - sir de 18 bytes neinitializati

41. **Cati biti are un cuvant format din 2 Bytes ?**

    16

42. **Cati biti de adresare sunt necesari pentru a accesa 128kb de memorie ?**

    17

43. **Cati bytes sunt intr-un cuvant dintr-un sistem cu memoria organizata pe 32 biti ?**

    4

44. **Care este utilitatea implicita a registrului DX?**

    Folosit in operatii input/output,aritmetice,cum ar fi inmultirea sau impartirea valorilor de dimensiuni mari

45. **Cate locatii de memorie pot fi selectate cu o magistrala de adrese pe 42 biti ?**

    4 TB

46. **Cate linii / biti de adresare sunt necesari pentru a accesa 32GB ?**

    32 GB = 2^35 => 35 de biti

47. **Cati bytes are o locatie de memorie daca avem un sistem pe 64 biti ?**

8 bytes

48. Ce rezulta din operatiunea OR pe biti 0101 | 1110 = ????

1111

49. Cum se numesc expresiile din programul ASM care incep cu punct si ce rol au ?

Expresiile se numesc puncte directive.Acestea nu creeaza nici un cod de masina ,deci nu contribuie la dimensiunea programului ; ele directioneaza asemblerul sa efectueze anumite actiuni in timpul fazei de asamblare.

ALTE RASPUNSURI // ASSEMBLER

AL DOILEA REFERAT*

.model small : Liniile care incep cu "***.***"reprezinta instructiuni speciale care indica programului assembler anumite informatii, descrise de cuvintele cheie ce urmeaza, cu privire la programul de construit.
.stack : Alta linie care descrie programul. Instructiunea indica locul in care incepe segmentul de stiva. Acesta este utilizat ca zona temporara de stocare a rezultatelor intermediare sau pentru a conserva starea sistemului (descrisa de valorile din registre) inainte de a efectua operatii care sa altereze datele existente.
.data : indica faptul ca incepe segmentul de date. Implicit, reprezinta terminarea segmentului de stiva. In segmentul de date sunt definite variabilele cu care se lucreaza in program.
a db 10: se defineste o variabila de tip byte cu valoarea zecimala 10.
.code : indica inceperea segmentului de cod si implicit, terminarea segmentului de date. Acest segment contine instructiunile programului.

mov AX, @data : Instructiunea incarca in registrul AX adresa segmentului de date. Alte simboluri predefinite @code – adresa segmentului de cod.

mov DS,AX : Instructiunea incarca in registrul segment DS adresa segmentului de date din registrul AX. Operatia este necesara deoarece nu este permisa incarcarea registrului DS cu o valoare constanta (adresa).

mov AL,a: se pune in registrul AL valoarea lui a.

add AL,b: se aduna la valoarea din registrul AL, valoarea lui b.

mov e,AL: variabila e este initializata cu valoarea din registrul AL

mov AX, 4c00h : incarca in registrul AX valoarea hexazecimala 4c00h. Acest lucru este necesar apelarii ulterioare a rutinei 21h.

int 21h : Apelul intreruperii 21h. In registrul AH se gaseste valoarea 4ch deoarece AX are valoarea 4c00h

.end **:** marcheaza sfârsitul fisierului sursa.

main proc : codul poate fi structurat prin intermediul procedurilor.

mesajdb "

`Afisare mesaj !!!"`: defineste o variabila sir de caractere numita *message* ce contine textul **`Afisare mesaj !!!.`**

mov AX, seg mesaj : Instructiunea incarca in registrul AX adresa segmentului de date. Reprezinta o alternativa la instructiunea *mov AX, @data* pentru ca instructiunea *seg* intoarce adresa segmentului in care se afla definita variabila respectiva.

mov AH, 09 : incarca in registrul AH valoarea constanta 09. Este necesar pentru a afisa un mesaj pe ecran utilizând intreruperea DOS 21h.

lea DX, message : Instructiunea LEA (Load Efective Address) incarca in registrul DX offset-ul din cadrul segmentului de date la care se gaseste variabila *mesaj* . Acest lucru este necesar pentru a afisa mesajul pe ecran utilizând intreruperea 21h.
int 21h : Instructiunea genereaza o intrerupere DOS. Procesorul apeleaza rutina indicata de numarul intreruperii, in acest caz o rutina DOS.

INSTRUCTIUNI

Instructiunea MOV

- realizeaza principalele operatii de transfer a valorilor;
- este atât de utilizata incât este imposibil sa scrii un program assembler fara aceasta instructiune;

Instructiunea XCHG

- interschimba valorile din *sursa* si *destinatie*;
- forma analitica:

Instructiunile LDS si LES

- sunt printre putinele instructiuni care prelucreaza un dublu cuvânt (32 de biti); transfera dublu-cuvântul din memorie catre 2 registre de 16 biti; valoarea din cei doi octeti superiori ai dublu-cuvântului este copiata intr-unul din registrele de segment (fie DS sau ES in functie de instructiune LDS sau LES); cuvântul (16 biti) mai putin semnificativ este copiat intr-un registru general indicat ca operand destinatie in instructiune; de obicei dublu-cuvântul este un pointer ce contine adresa unei variabile data de segment:offset (adresa segmentului pe 16 biti si offestul in cadrul segmentului tot pe 16 biti);
- incarca adresa in segment;
- forma analitica:

Instructiunea ADD

- aduna la operandul destinatie valoarea operandului sursa;
- adunarea se realizeaza pe 16 biti;
- forma analitica:

ADD *destinatie*, *sursa*

- permite multiple combinatii de tipuri ale sursei si destinatiei (registru, valoare imediata, locatie de memorie) insa exclude situatiile:

Instructiunea SUB (SUBstract)

- scade din operandul destinatie valoarea operandului sursa;
- forma analitica:

SUB *destinatie*, *sursa*

- restrictii identice ca la instructiunea ADD;

Instructiunea NEG

- utilizat pentru a nega operandul, scazând-ul din valoarea 0;
- forma analitica:

NEG *operand*

- registru FLAG afectat: OF,SF, ZF, AF, PF, CF;
- operandul trebuie sa fie registru sau variabila;
- când operandul contine valoarea minima posibila aceasta nu este afectata;

Instructiunea MUL (MULtiply)

- multiplica o valoare unsigned din AL (când operandul este de tip byte) sau din AX (când operandul este de tip word) cu valoarea operandului specificat; rezultatul este returnat in AX când operandul este de tip byte si in DX:AX (cei 2 octeti superiori in DX) când acesta este de tip word;
- forma analitica:

MUL *operand*

- registru FLAG modificat: OF si CF sunt setati daca partea superioara a rezultatului (DX daca operandul este de tip word, sau AH este de tip byte) este diferita de 0;alte flag-uri SF, ZF, AF, PF;
- operandul poate fi alt registru sau o variabila;
- daca valoarea inmultita are semn atunci se utilizeaza IMUL;

Instructiunea DIV (DIVide)

- imparte o valoare unsigned din AX (când operandul este de tip byte; in acest caz DX trebuie sa contina valoarea 0) sau din DX:AX (când operandul este de tip word) cu valoarea operandului specificat; câtul este returnat in AL iar restul in AH când operandul este de tip byte si in DX:AX (DX contine restul si AX câtul) când acesta este de tip word;

- forma analitica:

DIV *operand*

- registru FLAG modificat: nedefiniti;
- operandul poate fi alt registru sau o variabila;
- in cazul operandului de tip word memorarea deimpartitului in DX:AX se face cu instructiunea **cwd**(convert word to double) ce extinde valoarea din AX in zona DX:AX; exemplu:

Instructiunea ROR (ROtate Right)

- utilizata pentru a roti la dreapta bitii destinatiei de atâtea ori cât este specificat; cum fiecare bit este rotit la dreapta, cel mai putin semnificativ bit al destinatiei este copiat in locul celui mai semnificativ bit si in CF (carry flag);

Instructiunea ROL (ROtate Left)

- utilizata pentru a roti la stânga bitii destinatiei de atâtea ori cât este specificat; cum fiecare bit este rotit la stânga, cel mai semnificativ bit al destinatiei este copiat in locul celui mai putin semnificativ bit si in CF (carry flag);
- analogie cu ROR;

Instructiunea SAL sau SHL (Shift Arithmetic Left si SHift Left)

- se utilizeaza pentru a muta la stânga bitii din destinatie cu atâtea pozitii câte sunt specificate in contor; cu fiecare pozitie se adauga la stânga in bitul cel mai putin semnificativ valoarea 0;
- forma analitica:

SAL *destinatie*, *contor*

- registrul FLAG: SF, ZF si PF; cel mai semnificativ bit al destinatiei este mutat in CF; OF este setat daca contorul are valoarea 1 iar bitul de semn isi pastreaza valoarea;
- utilizat pentru a inmulti destinatia o putere a lui 2;
- destinatia poate fi registru sau variabila; contorul poate fi constanta sau registrul CL;

Instructiunea SAR (Shift Arithmetic Right)

- se utilizeaza pentru a muta la dreapta bitii din destinatie cu atâtea pozitii câte sunt specificate in contor; cu fiecare pozitie se adauga la dreapta in vecinul bitului cel mai semnificativ valoarea 0; bitul cel semnificativ (bitul de semn) isi pastreaza valoarea;
- forma analitica:

SAR *destinatie*, *contor*

- registrul FLAG: SF, ZF si PF; cel mai putin semnificativ bit al destinatiei este mutat in CF; OF este setat daca contorul are valoarea 1 iar bitul de semn si vecinul sau sunt diferiti;
- utilizat pentru a imparti un numar negativ (destinatia) la o putere a lui 2;
- destinatia poate fi registru sau variabila; contorul poate fi constanta sau registrul CL;

Instructiunea SHR (Shift Rigth)

- se utilizeaza pentru a muta la dreapta bitii din destinatie cu atâtea pozitii câte sunt specificate in contor; cu fiecare pozitie se adauga in vecinul bitului cel mai semnificativ valoarea 0;
- forma analitica:

Demultiplexoare. Multiplexoare

PRIMUL REFERAT*

(Asta in cazul in care ne va da sa rezolvam ,sa ne amintim cum se fac !!!)

Demultiplexorul (DMUX) este circuitul logic care distribuie datele de pe o cale de intrare pe mai multe căi de ieșire. Calea de ieșire pe care sunt transmise datele este selectată printr-un cuvânt de cod (adresă).

Circuitul are:

- *o intrare de condiționare a funcționării*, G_1, care permite funcționarea atunci cînd este legată la masă (nivel logic 0);
- *o intrare de date*, G_2, comună tuturor ieșirilor;
- ***n** intrări de adresă*;
- ***2^n** ieşiri active pe nivel logic 0*.

Un demultiplexor de 3 biți are 3 intrări de adresă (A,B,C) și 8 ieșiri ($0 \div 7$).

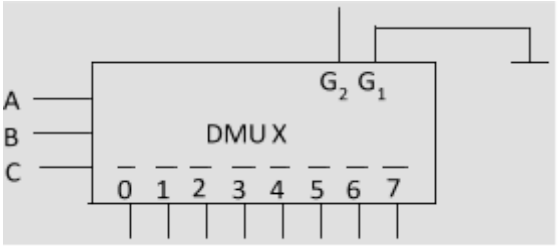


Fig.1 Schema bloc a unui DMUX de 3 biți

Funcționarea circuitului, ilustrată în tabelul de adevăr din Fig.2 pentru un demultiplexor de 2 biți, poate fi prezentată pe scurt astfel :

- dacă G_1 este pe nivel logic 1, circuitul este blocat, toate ieșirile fiind în 1 logic
- circuitul funcționează numai cu G_1 legată la masă
- pentru fiecare dintre cele 4 combinații distincte posibile care se pot aplica pe intrările de adresă este selectată câte o ieșire
- pe ieșirea selectată se transmit datele de pe G_2

G_1	G_2	A	B	0	1	2	3
1	x	x	x	1	1	1	1
0	0	0	0	0	1	1	1
0	1	0	0	1	1	1	1
0	0	0	1	1	0	1	1
0	1	0	1	1	1	1	1
0	0	1	0	1	1	0	1
0	1	1	0	1	1	1	1
0	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1

Fig. 2 Tabelul de adevăr al unui DMUX de 2 biți (4 căi)

Dacă nu se utilizează intrarea de date, demultiplexorul devine un decodificator cu rolul de a selecta ieșirea corespunzătoare codului binar aplicat pe intrările de adresă.

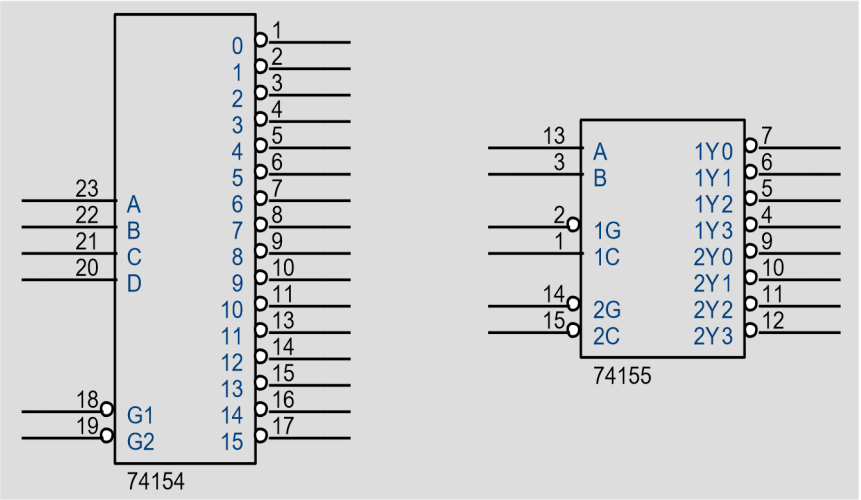


Fig. 3 Demultiplexoare integrate din seriaTTL

Multiplexorul (MUX) este circuitul logic care permite trecerea datelor de la una din intrări la o unică ieșire. Selecția intrării se realizează prin intermediul unui cuvânt de cod (adresă).

Circuitul are:

- *o intrare de condiționare a funcționării,* E, care permite funcționarea atunci cînd este legată la masă (nivel logic 0);

- ***n** intrări de adresă*;
- ***2**^{n} intrări de date*;
- *o ieşire*, W.

Un multiplexor de 3 biți are 3 intrări de adresă (A, B, C), 8 intrări de date ($x_0 \div x_7$) și o singură ieșire (W).

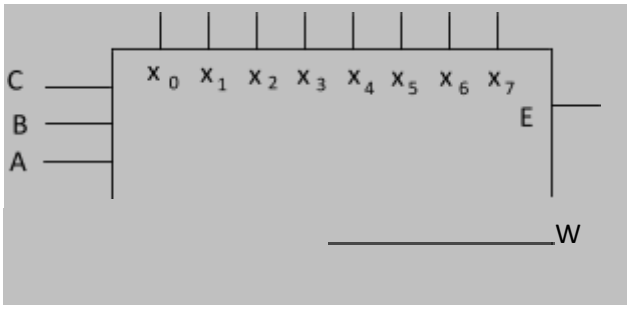


Fig.4 Schema bloc a unui MUX de 3 biți (8 căi)

Pentru un multiplexor de 2 biți funcționarea este ilustrată de tabelul de adevăr din Fig.5.

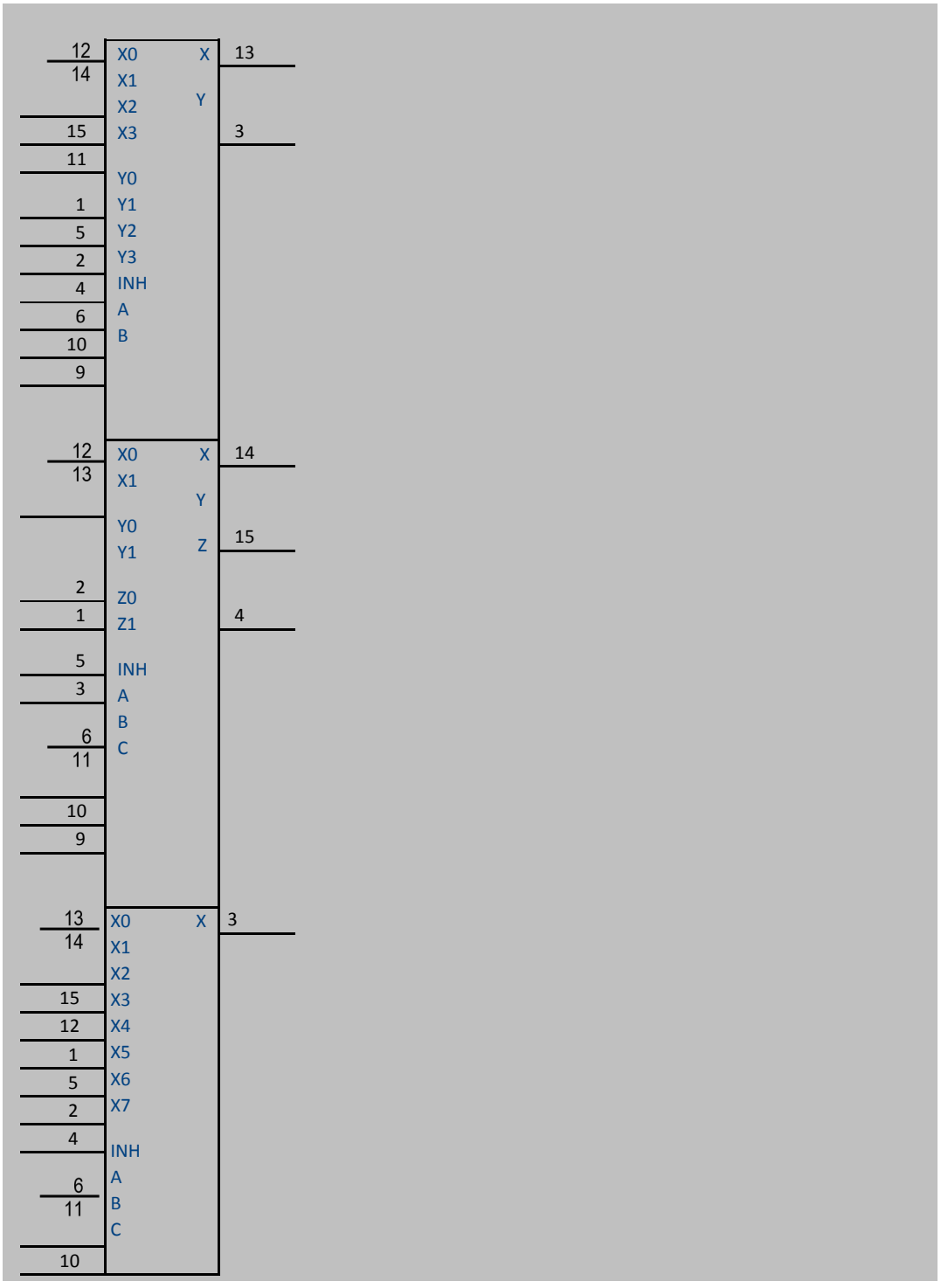
$\overline{E}$	A	B	x_0	x_1	x_2	x_3	W
1	x	x	x	x	x	x	0
0	0	0	0	x	x	x	0
0	0	0	1	x	x	x	1
0	0	1	x	0	x	x	0
0	0	1	x	1	x	x	1
0	1	0	x	x	0	x	0
0	1	0	x	x	1	x	1
0	1	1	x	x	x	0	0
0	1	1	x	x	x	1	1

Fig.5 Tabelul de adevăr al unui MUX de 2 biți (4 căi)

- Dacă $\overline{E}$ este pe 1logic, circuitul este blocat, ieșirea W fiind în 0 logic

- Cuvântul de cod aplicat pe intrările de adresă duce la selecția unei intrări de date Pe ieșire apar datele (0 sau 1) prezente pe intrarea de date selectată

Circuitele integrate multiplexoare pot avea pe lângă ieșirea W și ieșire $\overline{W}$, sau numai ieșire W.



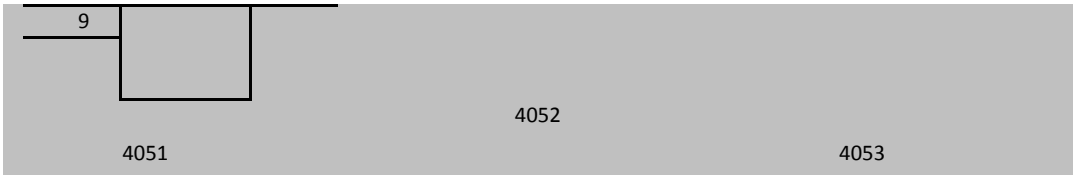


Fig.6 Multiplexoare integrate din seria CMOS

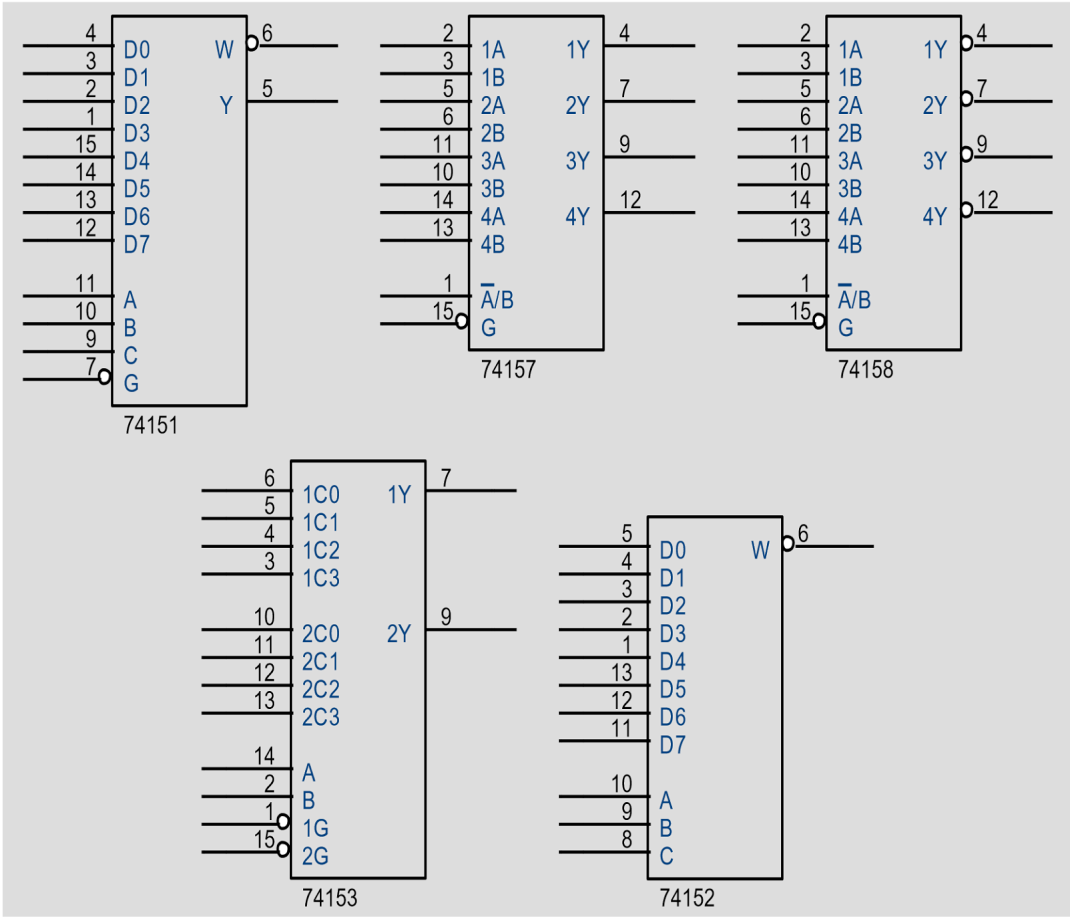


Fig.7 Multiplexoare integrate din seriaTTL

2	la puterea	1	=	2
2	la puterea	2	=	4
2	la puterea	3	=	8
2	la puterea	4	=	16
2	la puterea	5	=	32
2	la puterea	6	=	64
2	la puterea	7	=	128
2	la puterea	8	=	256
2	la puterea	9	=	512
2	la puterea	10	=	1024
2	la puterea	11	=	2048
2	la puterea	12	=	4096
2	la puterea	13	=	8192
2	la puterea	14	=	16384
2	la puterea	15	=	32768

Asta in cazul in care ne va intreba din puterile lui 2 vis -a-vis de linii / biti de adresare sunt necesari pentru …?