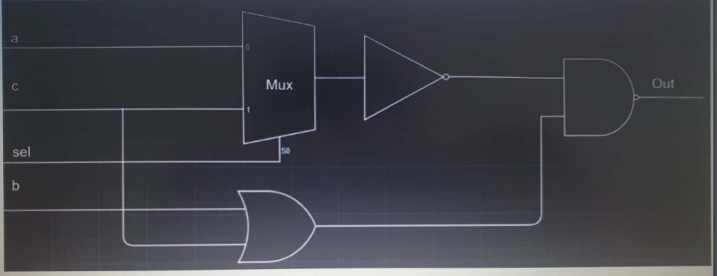
vr\_sa\_mor.exe

O imagine care conține masă

Descriere generată automat

BUNA RAU ASTA

1. Descrieti in Verilog folosind primitive si fara a folosi operatorii Verilog un modul care implementeaza diagrama din imagine.Pentru modulul mux se considera urmatoarea declarare : module mux(input in1, input in2 , output o , input sel)



REZOLVARE :

module modul(output out , input a , input , input sel, input b );

wire Mux\_out,OR\_out,N\_out;

mux Mux(a,c,Mux\_out,sel);

or OR1(OR\_out,b,c);

not NOT1(N\_out,Mux\_out);

nand NAND1(out,N\_out,OR\_out);

endmodule

2. Decodificati urmatoarea instructiune (exprimata in hexazecimal),tinand cont ca bitul 15 este aflat in stanga : 670a

REZOLVARE:

670a in binar , facut pe 16 biti dupa rasturnat rezultatul -> 0101 000 0 11 100 110

Ri0 = 0 , Ri1 = 1 , Ri2 = 0 Ri3 = 1

Instructiunea = 000 - >ADD

d = 0 -> R/M destinatie

->Reg = sursa

(Daca era d = 1 , R/M ->sursa si Reg ->Destinatie)

MOD = 11 => BA

R/M -> 110

Reg = 100 => 2^2 = 4 => XA ( pe asta il luam din SEL(schema calc didactic ) se Numara de sus in jos de la 0 )

Raspuns : ADD BA , XA

3. Implementati in Verilog un automat finit capabil sa recunoasca secvente de tip ’ab\*’ .Automatul primeste la intrare in mod continuu caractere codificate printr-un semnal de un bit ( caracterele posibile sunt ’a’ si ’b’).Iesirea automatului va consta dintr-un semnal care va fi activat ( valoarea 1 ) atunci cand la intrare am avut present un sir care se potriveste cu tiparul de cautare .

REZOLVARE :

https://cyberzhg.github.io/toolbox/nfa2dfa

module automat(output o,

input in,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 1;

else

next\_state=0;

end

1:begin

o=1;

if(i==0)

next\_state = 0;

else

next\_state = 1;

end

endcase

end

endmodule

4. Corectati erorile din urmatorul fragment de cod in Verilog care descrie un bistabil l de tip D.

//TOT DIN SARCINA EXERCITIULUI

module D\_flip\_flop(output reg Q , input D , clk,reset)

@posedge clk) //greseala

if(!reset)

Q = 0;//greseala

else

Q =D; //greseala

//dupa linie cu greseala nu e nici end la if

endmodule

REZOLVARE :

module D\_flip\_flop(output reg Q , input D ,clk,reset)

always @(posedge clk) begin

if(!reset)

Q<=0;

else

Q<=D;

end

endmodule

5. Ce rezultat are variabila rez , dupa executia codului de mai jos . Justificati

wire[1:0] a , b, rez;

assign a = ’b10;

assign b = 2 ;

assign rez = ~a>>>1 && b<<1

b = 2 => b = ’b10

a = ’b10

~a = 01

~a >>>1 = 00

b<<1 => b = 00

rez = 00 && 00 = 00; //ATENTIE LAPRIOTITATI

6. Codificati instructiunea mov [XB] , RC

RC pe pozitia care e => d = 0;

RC e pe poz 2 in schema de la calc didactic => REG = 010

Rm adica [XB] 101

MOD pt [XB] 00

//si acm futai

pt MOV stim sigur

INSTRUCTIUNEA MEREU 000 PE ORICE CAZ

ca RI0 = 0;

si mai stim sigur ca RI1 = 0

si RI3 = 0; ca sa fie numai alea de sus de tot

RI2 = 0 ( creca e fara operand imediat )

deci vine

RASPUNS :

0000 000 0 00 010 101

7. Codificati instructiunea SAR [XA +1]

Raspuns: 0001 110 x 10 xxx 110

RI1=0; RI2=0;RI3=0;RI4=1

SAR=110

[XA+1] se afla in tabelul de mai jos pe coloanal 10 si linia 100

Restul nu conteaza

9. Codificati instructiunea DEC[BB+XB-3]

Raspuns: 0001 001 x 10 xxx 011

RI1=0; RI2=0;RI3=0;RI4=1

SAR=001

[BB+XB-3] se afla in tabelul de mai jos pe coloana 10 si linia 011

Restul nu conteaza.

11. Implementati in Verilog un automat finit capabil sa recunoasca secvente de tip ’ba’

module ba(

output reg o, // found output: 0 - not found, 1 - found

input i, // char input: 0 - 'a', 1 - 'b'

input clk); // clock input

//TODO implementarea functionarii

reg [1:0] state = 0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always @(\*) begin

o = 0;

case(state)

0: begin

if(i == 0)

next\_state = 0;

else

next\_state = 1;

end

1: begin

if(i == 0)

next\_state = 2;

else

next\_state = 1;

end

2: begin

o = 1;

if(i == 0)

next\_state = 0;

else

next\_state = 1;

end

endcase

end

endmodule

12. Ce stocheaza si cand e folosit registrul T1?

Registrul temporar T1 este utilizat pentru a păstra operanzii unei operaţii executate în UAL, rezultate intermediare la calcularea adresei efective şi nu sunt accesibile în mod explicit de programator.

In procesul de prelucrare, operanzii sunt preluaţi din RG şi transferaţi în registrele de lucru T1, T2 ca intrări pentru unitatea aritmetică logică UAL

13. Ce reprezinta si la ce se foloseste bitul D (eng. O) din cadrul registrului de indicatori?

indicatorul de depăşire D este setat dacă în urma execuţiei unei operaţii aritmetice rezultatul este un număr pozitiv prea mare sau un număr negativ prea mic pentru a putea fi reprezentat în operandul destinaţie (exclusiv bitul de semn); altfel D este şters.

poate fi interpretat ca depăşire în operaţiile cu numere întregi cu semn şi poate fi testat cu instrucţiuni de salt condiţionat.

poate fi ignorat în operaţiile aritmetice cu numere întregi fără semn.

14. Codificati instructiunea PUSH[BB+XB+10]

Raspuns: 0000 010 X 10 XXX 011

RI1=0; RI2=0;RI3=0;RI4=0

PUSH=010

[BB+XB+10] se afla in tabelul de mai jos pe coloana 10 si linia 011

Restul nu conteaza.

15. Codificati instructiunea POP[BA+XA+]

Raspuns: 0000 011 X 01 XXX 000

RI1=0; RI2=0;RI3=0;RI4=0

POP=011

[BA+XA+] se afla in tabelul de mai jos pe coloana 01 si linia 000

Restul nu conteaza

15. Codificati instructiunea PUSH[RB] : NU EXISTA

16. Codificati instructiunea AND RC,[BA+XB-]

Adresare indirectă prin sumă de registre cu autodecrementare a registrului index înainte de calculul adresei efective.

Raspuns: nu exista pentru ca nu exista [BA+XB-]

17. Codificati instructiunea MOV [XB],RC

Raspuns: 0000 000 0 00 010 101

RI0=0; RI1=0;RI2=0;RI3=0

MOV = 000

d=0

MOD = 00

RC se afla pe linia 010

[XB] se afla pe linia 101

18. Ce stocheaza si cand e folosit registrul AIE?

Registrul de adrese AIE va fi folosit pentru a stoca adresa portului cu care se dorește să se comunice. Spre exemplu, atunci când procesorul vrea să transmită ceva pe interfața paralelă, el va depune în registrul AIE valoarea 0x378 (ce identifică portul asociat interfeței paralele), va activa pe magistrală cuvantul de date ce se dorește a fi transmis, iar unitatea de comandă va lansa semnalul «I/O Write» ce va determina încărcarea cuvântului în registrul interfeței paralele.

19. Descrieti 2 moduri de adresare folosite de calculatorul didactic si justificati necesitatea lor.

1. Adresare directa la registru: Operandul se gaseste in RG.

Exemplu: MOV RA,RB -> Instrucțiunea are ca efect încărcarea în registrul RA a valorii din registrul RB.

2. Adresare imediata: Operandul este specificat în instrucțiune.

Exemplu: MOV RA, 7- > Instrucțiunea va avea ca efect încarcarea valorii “7” în registrul RA. “7” poartă numele de operand imediat.

Modurile de adresare reprezintă modalitatea prin care se poate specifica adresa efectivă a operanzilor. Modurile de adresare permise de calculatorul didactic sunt tipice arhitecturilor CISC, fiind derivate din arhitectura standard x86. În total, procesorul permite 11 moduri de adresare, ceea ce îi conferă o foarte bună flexibilitate în programare.

20. Codificati instructiunea: XOR [BA+XA+], RB

Raspuns: 0101 110 0 01 001 000

RI0=0; RI1=1; RI2=X (nu stim daca e cu/fara operand imediat); RI3 = 1

d = 0

MOD = 01

RB se afla pe linia 001

[BA+XA+] se afla pe linia 000

20. Se transforma din hexazecimal in binar pe 16 biti: 0318

0000 0011 0001 1000

In rastorn: 0001 100 0 11 000 000

Caut in tabel codificarea: SHL RA

21. Care este diferenta intre indicatorul de transport si cel de depasire? Dati exemple care sa exemplifice utilizarea lor.

Indicatorul de transport se activeaza (devine 1) in momentul in care adunam 2 numere si regultatul estev pe mai multi biti decat cele 2 numere.

Indicatorul de depasire se activeaza (devine 1) in momentul in care se aduna 2 numere negative si se obtine rezultate pozitiv sau in momentul in care se aduna 2 numere positive si se obtine un numar negative.

Exemple:

0001 1100 0100 1000

0001 0100 0100 1000

------- ------- ------- --------

0010 10000 1000 10000

t=0 t=1 t=0 t=1

d=0 d=0 d=1 d=1

22. Se transforma din hexazecimal in binar pe 16 biti: 273a

0010 0111 0011 1010

Il rastorn: 0101 110 0 11 100 100

Caut in tabel codificarea: XOR XA,XA

23. Se transforma din hexazecimal in binar pe 16 biti: 1080

0001 0000 1000 0000

Il rastorn: 0000 000 1 00 001 000

Caut in tabel codificarea: MOV RB, [BA+XA]

24. Se transforma din hexazecimal in binar pe 16 biti: 218a (fara 0001)

0010 0001 1000 1010

Il rastorn: 0101 000 1 10 000 100

Caut in tabel codificarea si transform 0001 de la final in decimal(o sa fie deplasarea): ADD RA, [XA+1]

25. Se transforma din hexazecimal in binar pe 16 biti: e304 (fara 0080)

1110 0011 0000 0100

Il rastorn: 0010 000 0 11 000 111

Caut in tabel codificarea si transform 0080 de la final in decimal: MOV BB, 128

26. Se transforma din hexazecimal in binar pe 16 biti: 1080

0001 0000 1000 0000

Il rastorn: 0000 000 1 00 001 000

Caut in table codificarea: MOV RB, [BA+XA]

26. Se transforma din hexazecimal in binar pe 16 biti: 5a00

0101 1010 0000 0000

Il rastorn: 0000 000 0 01 011 010

Caut in table codificarea: MOV [BB+XB+], RC

----------- Aici s au oprit eroii urmatori : Adam , Azoitei , Hamzu , Nomolosanu , Seuleanu ------

----------------------------------------------- Dupa au cazut la datorie -----------------------------------------

27. Descrieti in Verilog folosind primitive si fara a folosi operatorii Verilog un modul care implementeaza diagrama din imagine . Pentru modulul dmux se considera urmatoarea declarare : module dmux ( input in , output o1 , output o2 , input sel);

O imagine care conține text, tablă, cer de noapte

Descriere generată automat

Rezolvare:

module my\_module(output o , input [2:0] I ,input sel);

wire out,out2,out3,in\_demux

not NOT(out , i[1]);

or OR(out2,out , i[2]);

nand NAND(out3,i[0],i[1]);

nor NOR(in\_dmux ,out2,out3);

dmux(in\_demux,o[0],o[1],sel[0];

endmodule

28.Cu cati ciclii dureaza maim ult instructiunea sbb RA,7 decat instructiunea sbb RA,RB? Justificati fiecare ciclu .

//cnv ?

28. Descrieti pe scurt executia instructiunii cmp XA,[XA+1] , de la citirea ei din memorie pana la trecerea la instructiunea urmatoare .Care sunt pasii urmati de unitatea de comanda , ce componente ale procesorului sunt implicate in prelucrarea ei si cum sunt folosite acestea ?

Vedem ca trebuie sa comparam XA cu XA+1 .

Etapa de fetch : cp oe , scriem pe AM we

In al 2 lea pas facem AM oe , asteptam un ciclu de ceas

Etapa 3 : RI we si RAM oe .

Etapa 4 : decode

Modul de adresare : 10

COP [0:2] 010 cmp

Executia cu 2 operanzi

1:Calculam adresa lui XA+1 , incarcam pe T1 XA

2:Setam operatia ALU COD ADC

3:Vrem sa salvam in AM ALU COD OR si Salvam in AM

AM <= T1|0

4: Facem AM oe = 1 (cautam in memorie)

5: T2 <= M(AM) (sau M(XA+1)

6: T1 <= XA;

7: ALU COD = sbb1

8: Increment CP

9: Se seteaza flag urile .

29.Considerand implementarea calculatorului didactic , scrieti in Verilog starile complete pentru executia urmatoarei operatii : T2<-M[CP].

//similar fetch

CP oe = 1 , AM we = 1

1. asteptam un ciclu de ceas

-facem AM oe = 1 .

3.Facem RAM oe = 1 , T2 we .

//sau (mrg doar asa )

CP oe = 1 si T2 we = 1

(Alu op code OR , facem T2 oe = 1

ALU oe = 1) circuit combinational , pe magistrala se afla output ul de la ALU , AM we = 1

pe care-l scriem in AM .

Cautam in memorie AM oe = 1 , asteptam un ciclu de ceas .

Facem T2 we = 1 si in T2 avem M(CP).

30.Descrieti pe scurt executia instructiunii dec RC , de la citirea ei din memorie pana la trecerea la instructiunea urmatoare . Care sunt pasii urmati de unitatea de comanda, ce componente ale procesorului sunt implicate in prelucrarea ei si cum sunt folositeacestea ?

Etapa fetch

1 : CP oe =1 AM we = 1

2: AM oe = 1

3:RAM oe = 1 RI we = 1

etapa decode :

DEC , un operand , mod de adresare 11 => next EXECUTE

RC oe = 1 T1 = we ;

in T1 <= RC

2.T1 oe ALU OPCODE =SBB1(carry1 )

ALU oe = 1

T1 we = 1 .

3: T1 <= T1-1 ;

4:STORE : T1 oe = 1 ,ALU OPCODE OR

ALU oe =1; RC we =1;

5 INC CP

31. Implementati in Verilog un automat finit capabil sa recunoasca secvente de tip ’bab’ .Automatul primeste la intrare in mod continuu caractere codificate printr-un semnal de un bit ( caracterele

//automat = bab

module automat(output reg o,

input i,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 0;

else

next\_state=1;

end

1:begin

if(i==0)

next\_state = 2;

else

next\_state = 1;

end

2:begin

if(i==0)

next\_state = 0;

else

next\_state = 3;

end

3:begin

o=1;

if(i==0)

next\_state = 0;

else

next\_state = 1;

end

endcase

end

endmodule

32. Cu cati ciclii dureaza mai mult instructiunea sbb RA,7 decat instructiunea RA,RB ? Justificati fiecare ciclu in plus .

Dureaza mai mult cu 4 ciclii , pt ca incrementarea cp ului dureaza 2 ciclii ,si luatul din memorie dureaza 3 ciclii.(7 e operand imediat) fata de ra,rb dureaza 4 ciclii (

SBB RA, 7 - T1 <- RA, T2 <- CP + 1, CP <- T2, AM <- CP, citire AM, T2 <-RAM[AM], T1 <- T1 - T2, RA <- T1

SBB RA, RB - T1 <- RA, T2 <- RB, T1 <- T1 - T2, RA <- T1)

33. Automat ‘bbb’

//automat = bbb

module automat(output reg o,

input i,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 0;

else

next\_state=1;

end

1:begin

if(i==0)

next\_state = 0;

else

next\_state = 2;

end

2:begin

if(i==0)

next\_state = 0;

else

next\_state = 3;

end

3:begin

o=1;

if(i==0)

next\_state = 0;

else

next\_state = 3;

end

endcase

end

endmodule

34.Cu cati ciclii dureaza mai mult instructiunea and[BA+XB+3],RA decat instructiunea and BA,RA?

Justificati fiecare ciclu in plus .

AND [BA + XB + 3], RA => T1 <- BA, T2 <- XB, T1 <- T1 + T2, CP <- CP + 1, T2 <- RAM[CP], T1 <- T1 + T2, T1 <- RAM[T1], T2 <- RA, T1 <- T1 && T2, RAM <- T1 = 1+1+1+2+3+1+3+1+1+2 = 16

AND BA, RA => T1 <- BA, T2 <- RA, T1 <- T1 && T2, BA <- T1 = 1+1+1+1 = 4

35. De cate ori se modifica registrul de indicatori in cadrul executiei instructiunii mov RB,[BA+XB+1]? Justificati.

Raspuns: Pentru ca avem mov, nu se modifica registrul de indicatori, deci de 0 ori.

36. //automat =’ aab’

module automat(output reg o,

input i,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 1;

else

next\_state=0;

end

1:begin

if(i==0)

next\_state = 2;

else

next\_state = 0;

end

2:begin

if(i==0)

next\_state = 0;

else

next\_state = 3;

end

3:begin

o=1;

if(i==0)

next\_state = 3;

else

next\_state = 3;

end

endcase

end

endmodule

37. Descrieti in Verilog folosind primitive fara a folosi operatorii Verilog un modul care realizeaza urmatoarea funtie logica : f(x,y,z) = (x|y)&(x| !y) & (x^z) . x,y,z sunt intrari pe 1 bit.

module nume(output out,input x , input y , input z );

wire y1,y2,y3;

and(out,y1,y2,y3);

or(y1,x,y);

or(y2,x,not y);

xor(y3,x,z);

endmodule

38. Descrieti in Verilog folosind primitive si fara a folosi operatorii Verilog un modul care realizeaza urmatoarea functie logica: f(a,b,c,d)=(a^b)+(c&d)+d, a,b,c,d sunt intrari pe 1 bit.

module modul(output out, input a,b,c,d);

wire out1,out2;

xor XOR(out1,a,b);

and AND(out2,c,d);

or OR(out1, out2, out , d);

endmodule

39. //automat =’ a+b’

module automat(output reg o,

input i,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 1;

else

next\_state=0;

end

1:begin

if(i==0)

next\_state = 1;

else

next\_state = 2;

end

2:begin

if(i==0)

next\_state = 2;

else

next\_state = 2;

end

end

endcase

end

endmodule

40. //automat = ba+

module automat(output reg o,

input i,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 0;

else

next\_state=1;

end

1:begin

if(i==0)

next\_state = 2;

else

next\_state = 1;

end

2:begin

if(i==0)

next\_state = 2;

else

next\_state = 0;

end

end

endcase

end

endmodule

41. //automat = aaa

module automat(output reg o,

input i,//0=a,b=1

input clk

);

reg [1:0] state=0;

reg [1:0] next\_state;

always @(posedge clk) begin

state <= next\_state;

end

always@(\*) begin

o=0;

case(state)

0:begin

if(i==0)

next\_state = 1;

else

next\_state=0;

end

1:begin

if(i==0)

next\_state = 2;

else

next\_state = 0;

end

2:begin

if(i==0)

next\_state = 3;

else

next\_state = 0;

end

3:begin

o=1;

if(i==0)

next\_state = 1;

else

next\_state = 0;

end

endcase

end

endmodule

42. Corectati erorile de sintaxa din urmatorul fragment de cod Verilog.

module my\_mod(output out, input b, input c)

wire nb;

always @(\*) begin

not(nb,b)

a=nb&c;

end

assign out =a;

endmodule

Raspuns:

module my\_mod(output out, input b, input c);

wire nb;

not(nb,b);

always @(\*) begin

a=nb&c;

end

assign out =a;

endmodule

43. Etapa de fetch

Etapa de fetch presupune aflarea adresei instructiunii pe care o voi executa .

Are 3 stari ->1. cp -> oe = 1 ;

am\_we = 1 ;

starea 2. am oe = 1 ;

starea 3 .ram\_oe = 1 , ri we = 1 ;

43. Descrieti pe scurt executia instructiunii XOR RA,IS de la citirea ei din memorie pana la trecerea la instructiunea urmatoare .Care sunt pasii urmati de unitatea de comanda , ce componente ale procesorului sunt implicate in prelucrarea ei si cum sunt folosite acestea ?

Din CP luam adresa cu oe si dam am\_we pentru a putea cauta in memorie , dupa ce am gasit instructiunea codificata in memorie , plecam pe magistrala catre RI , in RI decodificam instructiunea , trecem in etapa de decodificare , apoi incarcam valorile din registrii in T1(RA) si T2(IS). Intram in UAL , se face XOR-ul ,apoi salvam “rezulatul” in T1 , apoi se salveaza rezultatul in RA iar la final INC CP.

44. Considerand implementarea Calculatorului Didactic , scrieti in Verilog starile complete pentru executia urmatoarei operatii RC<-M[IS].

s1:

IS\_oe = 1 ; // iei adresa lui IS

AM\_we = 1 ;//dai we ca sa pot isa bagi adresa lui IS in AM

next\_state = s2;

end

s2 :

AM\_oe= 1 ;

next\_state = s3 ;

end

s3:

M\_oe = 1;

RC\_we = 1;

next\_state s4;

end

AM<-IS //in AM punem ADRESA LUI IS

Se citeste AM

RC<=M[AM]

45.De cate ori se modifica registrul de indicatori in cadrul executiei instructiunii CMP RB, [BA+XB].Justificati.

Raspuns : Nu trece prin ALU , deci nu se modifica niciun flag.

/\*46.De cate ori se modifia registrul de indicatori in cadrum executiei instructiunii ADD RA,[BA+XA+2] \*/

Se modifica.

47 . Codificati instruiunea : push [RB]

raspuns : N/A nu exista !

48. Decodificati instruciunea :

0101100|0|10|000|010|0000000000000101|

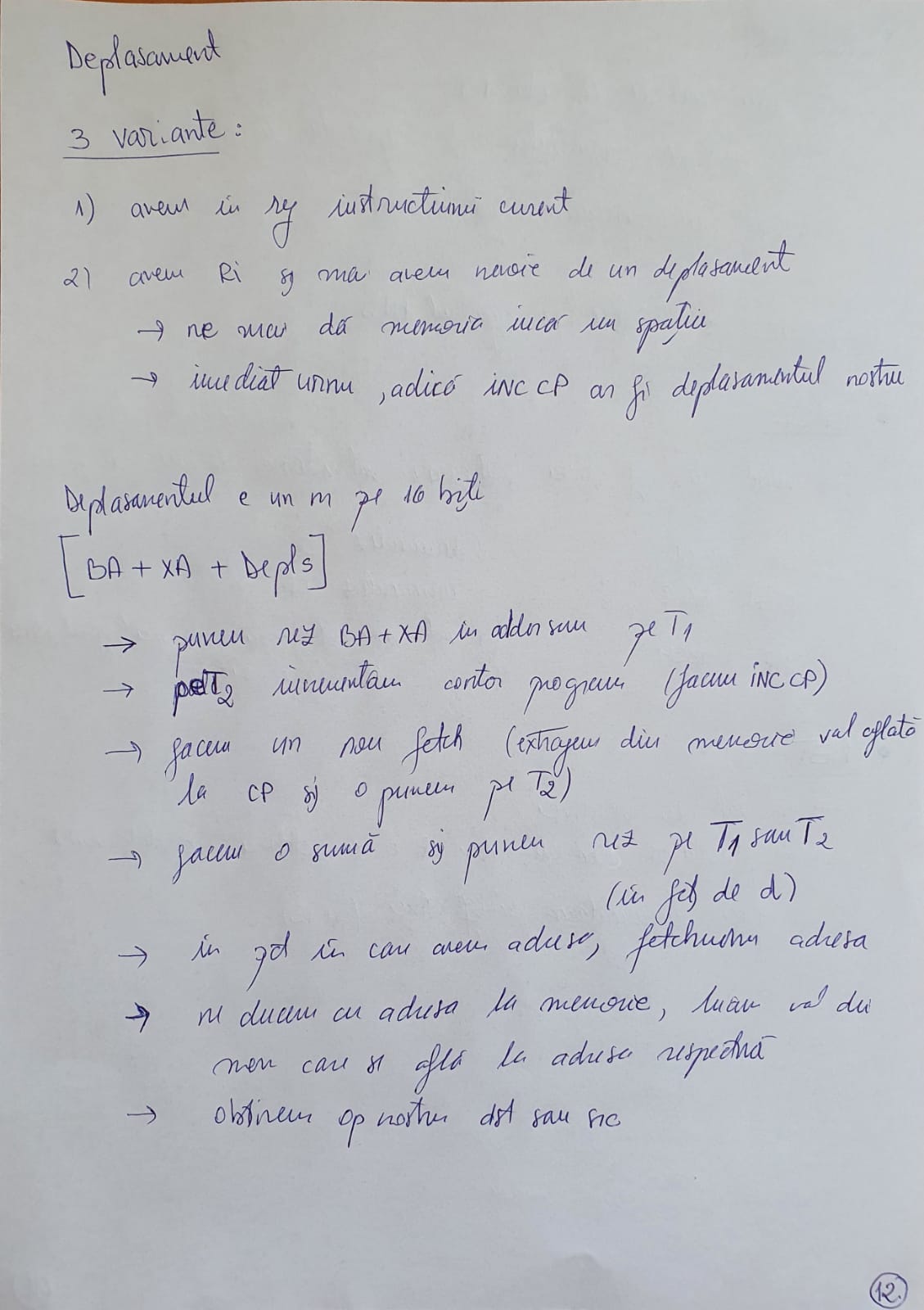
Raspuns : and [BB+XA+5],RA

49. Codificati instructiunea : adc [BB+XB+5],10

raspuns : 0101001|x|10|xxx|011|0000000000000101|0000000000001010

50. Decodificati instruciunea 0101110|0|10|001|000|0000000000000111|

Raspuns : xor [BA + XA + 7],RB



// buna asta x 2

51. ADD [BA+XA+2], 6 practicc cum se executa asta

T1<= ba

T2<= xa

T1<=T1+T2

T2<=cp

cp<=T2+1

am<=cp

asteptam am

t2<=ram

T1<=T1+T2

am<=T1

asteptam am

T1<=m(am)

2<=cp

cp<=T2+1

am<=cp

asteptam am

T2<=ram

T1<=T1+T2

.....

store:

T1\_oe=1

alu op\_code = or

alu\_oe=1

am\_oe=1

ram\_we=1

CMP BB,[XA]

O imagine care conține text

Descriere generată automat

ADC [BA+XA+2],BA

O imagine care conține text, tablă albă

Descriere generată automat

SUB [BB],3

O imagine care conține text, tablă albă

Descriere generată automat

INC RA

INC [BB+XB]

O imagine care conține text

Descriere generată automat