Sisteme logice secventiale complexe

Unitate de executie
Unitate de comandă
Microprogramare

S.l. Dr. Ing. Vlad-Cristian Miclea

Universitatea Tehnica din Cluj-Napoca
Departamentul Calculatoare

CUPRINS

PLAN CURS

- Partea 1 – VHDL
  1. FPGA
  2. Limbajul VHDL – 1
  3. Limbajul VHDL – 2
  4. Limbajul VHDL – 3
- Partea 2 – Implementarea sistemelor numerice
  5. **Realizarea unui sistem numeric complex; Unitate de executie**
  6. **Unitate de comanda; Microprogramare**
- Partea 3 – Automate
  7. Automate finite
  8. Stari
  9. Automate sincrone
  10. Automate asincrone
  11. Identificarea automatelor
  12. Automate fara pierderi
  13. Automate liniare
- Partea 4 – Probleme si discutii

CONTEXT

Sisteme numerice sincrone complexe

- Realizarea unor sisteme digitale complexe
- Se va discuta abordarea, definirea, analiza, implementarea si testarea onor sisteme hardware complexe
- Exemplu: un cuptor de gatit
- Se va discuta realizarea componentelor principale: UC si UE
- Unitate de executie
  - Numaratoare, registre, MUX-uri, ALU
  - Trebuie accesate la momentul potrivit
- Unitate de comanda/control
  - Generare semnale de control
  - Generarea urmatoarei stari
  - Unitate cablata (hardwired)
  - Microprogramare

Definirea problemei

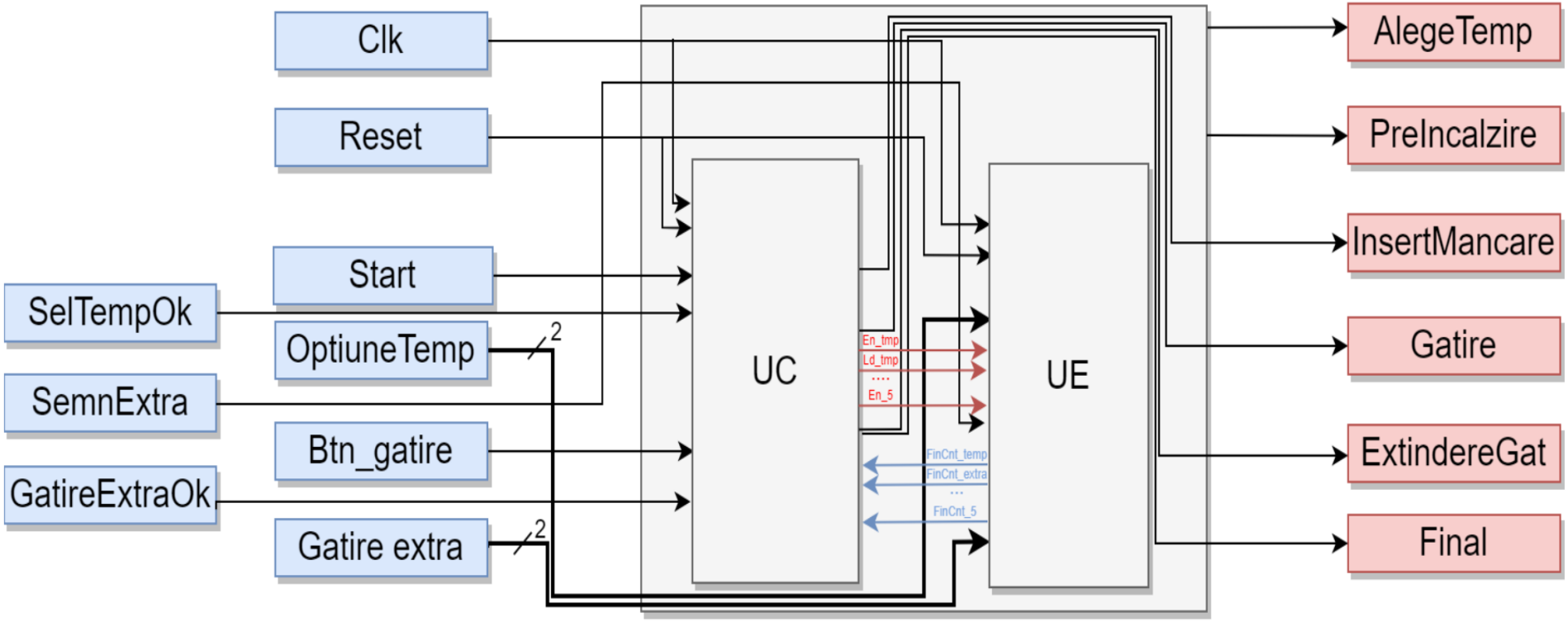
- Cuptor de gatit – exemplul de la laborator (putin mai complicat)
  - Gateste la 4 temperaturi predefinite: 180, 200, 220, 240 grade
    - In fiecare secunda, temperatura creste cu 10 grade;
  - Gateste 25 de minute
  - La sfarsit, se poate extinde perioada cu o valoare de baza de 50 de minute, la care se poate aduna sau reduce 5, 8 sau 12 min fata de perioada de gatire extra
- Detalii:
  - Se asteapta pana cand se apasa buton “Start”
  - Daca Start, se asteapta alege temp – se asteapta introducerea temp (V0,V1,V2,V3);
  - Se preincalzeste cuptorul; Apare un led “Preincalzire”
  - Apoi se stinge “Preincalizre”, se aprinde “InsertMancare” (se sta maxim 5 min)
  - Daca se introduce mancarea, se apasa “Buton_gatire” si se asteapta 25 min (se fiseaza “Gatire”)
  - Dupa finalizare gatire, se asteapta extindere gatire
  - Daca se doreste extra-gatire, se alege optiunea dorita
  - Se calculeaza timpul de extra-gatire
  - Se asteapta timpul de gatire suplimentare; Apare un led “Extra_gatire”
  - La final, se stinge ledul “Gatire” si Extra-gatire si se asteapta un nou proces

UC pentru cuptor – C6

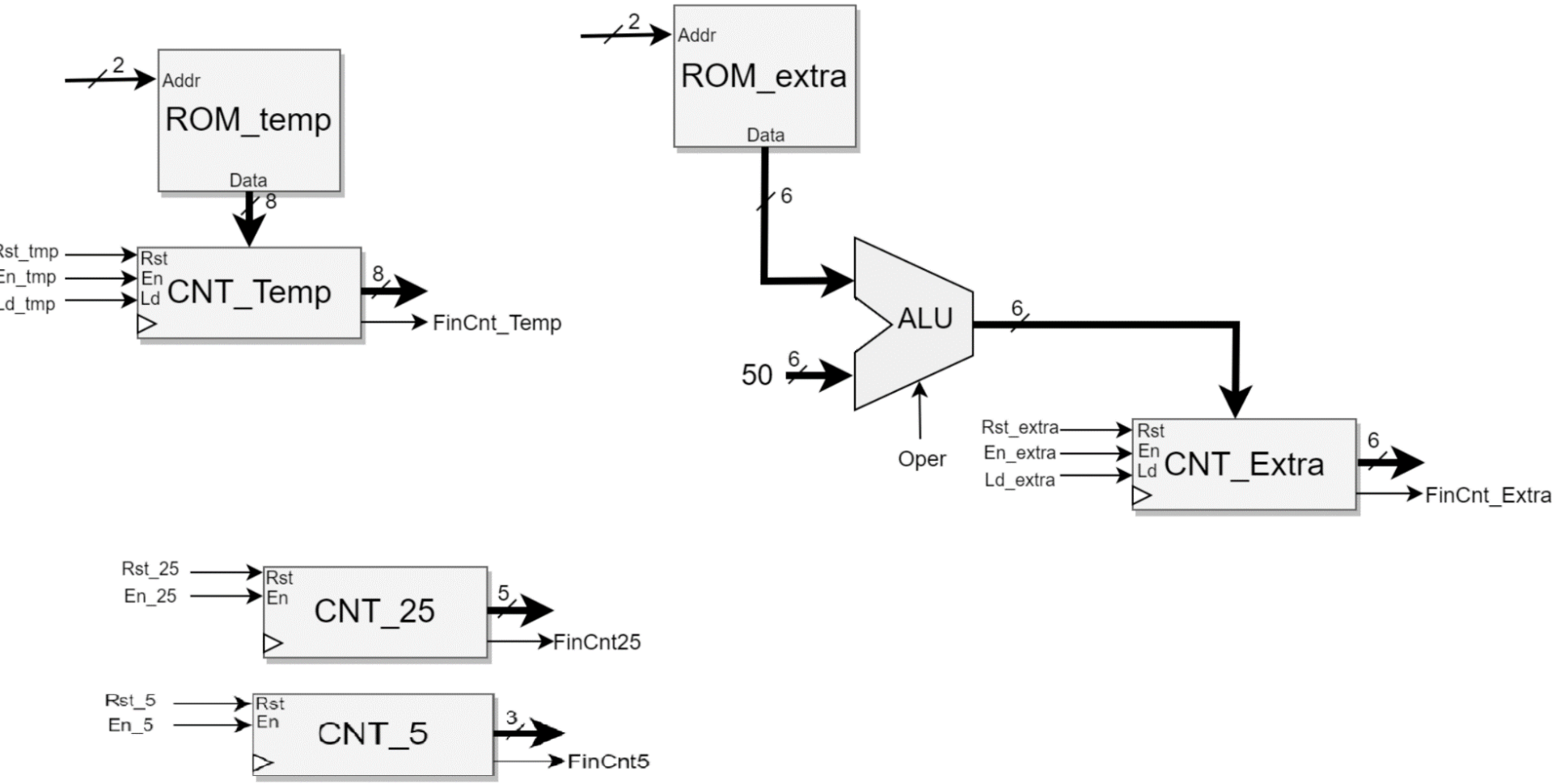
- Circuit = Unitate Control + Unitate Executie
- Unitate Control – Cablata sau microprogramata
- Unitatea microprogramata
  - Foloseste o memorie
  - Tine semnalele de control si microinstructiunea urmatoare
  - Microoperatii – genereaza semnalele de control pt UE
- Astazi: UC cablata + UC microprogramata pentru cuptor

Cuptor – UC vs UE

- Apar semnalele de control
  - De la UC spre UE
  - Rol de enable, load, reset sau de transmisie date
- Semnalele de raspuns
  - De la UE spre UC
  - Finalizare numarare

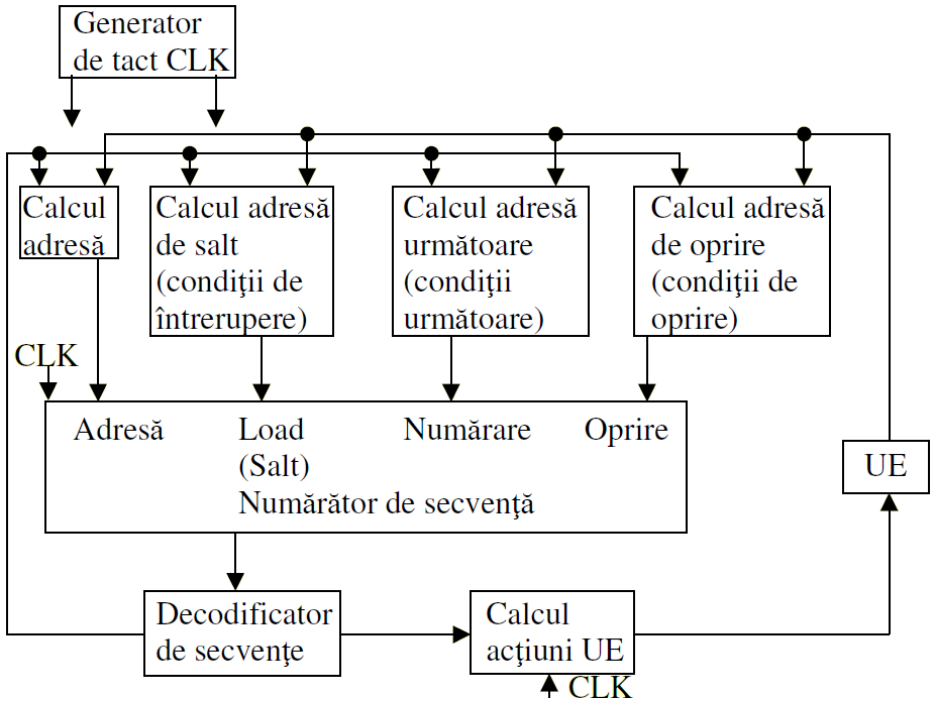


SCHEMA UE - CUPTOR



Unitate cablata (hardwired)

- Foloseste un FSM (finite state machine) -
  - Contine o parte combinationala – genereaza urmatoarea stare
  - Contine un registru/numarator – tine starea curenta
  - Fiecare stare corespunde la un set de semnale de control
- Metoda rapida, toate informatiile se genereaza direct din codificarea starii
- Utilizare limitata (circuite simple)
- Greu de extins cu semnale noi
- Greu de extins cu stari noi
  - Trebuie redefinit tot sistemul
- Exemplu



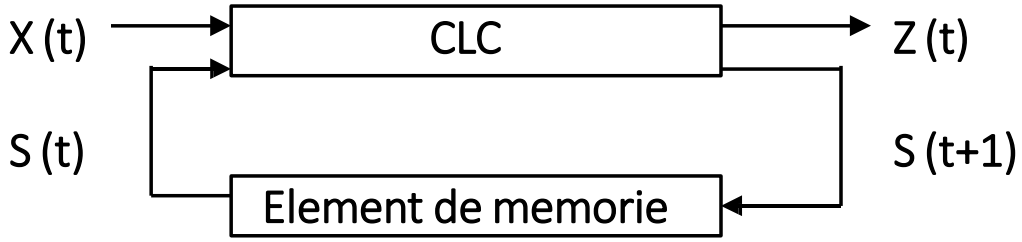
FSM (automat de stari)

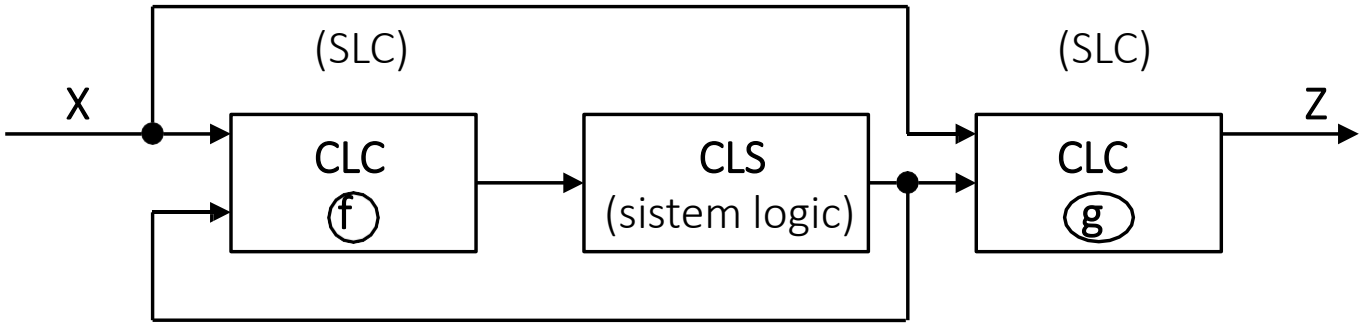
- Model de calculabilitate, folosit pentru proiectarea unor circuite secvenţiale
- Ajuta sa modelam execution flow-uri
- Una din aceste stari poate fi activa in oricare moment de timp
  - trebuie sa isi schimbe starea activa (sau curenta) in functie de niste conditii prestabilite
- Pot primi intrări şi pot da la ieşire diverse informaţii
- Au 2 componente:
  - Componenta secventiala –un registru/numarator -> genereaza starea urmatoare la momentul de timp definit (pe ceas)
  - Componenta combinationala – stabileste starea urmatoare
    - Stabileste si iesirile in starea curenta
    - Tine cont de intrarile externe ale circuitului
- Pot modela direct organigrame (ex semafor lab)

FSM (automat de stari)

Schema bloc

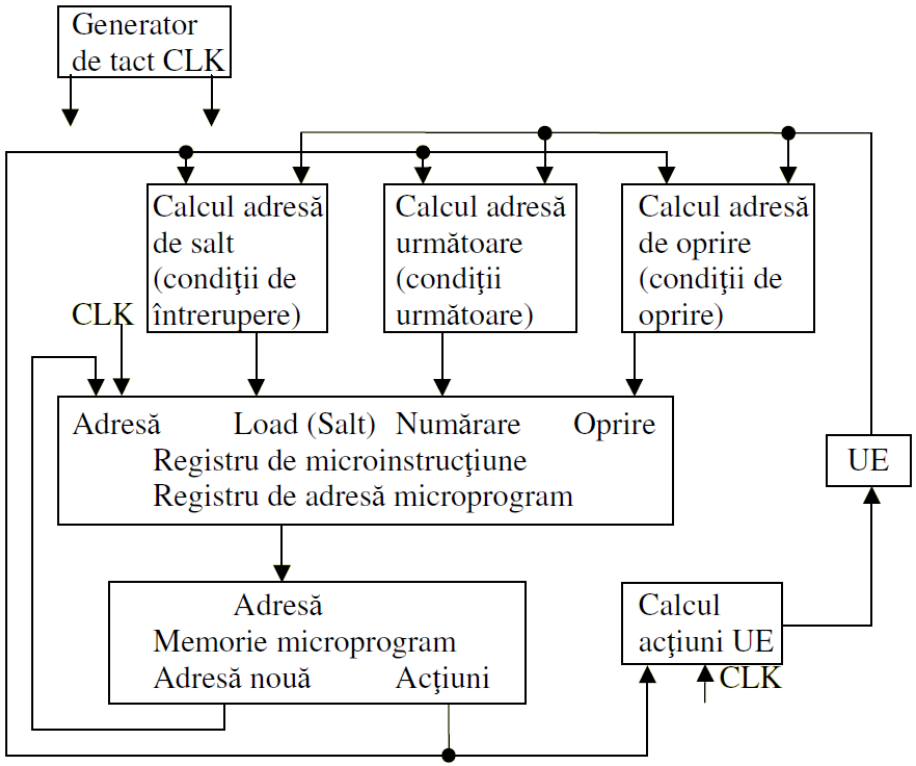
- 2 variante





Unitate microprogramata

- Semnalele de control nu mai sunt codificate intr-o anumita stare
- Secventa de instructiuni va fi generata folosind o **memorie**
- Registrul va incara o microinstructiune, care va contine o **adresa**
- Datele de la adresa respectiva vor contine informatii pentru a genera:
  - Semnalele de control
  - Adresa urmatoarei microinstructiuni



UNITATEA DE COMANDA/CONTROL

Sinteza unităţii de comandă UC

- realizarea microprogramată a unităţii de comandă a unui sistem numeric se bazează pe utilizarea unei memorii ROM
- cuvintele memoriei reprezintă fiecare o microinstrucţiune a programului
- în principal există cuvinte de test, de adresă şi de comandă
- se defineşte o metodă de sinteză pentru UC a sistemelor numerice

METODA DE SINTEZĂ

Sinteza unităţii de comandă UC

- o UC microprogramată este de fapt o unitate de tratare a adreselor din memoria de microprogram
- metoda de sinteză a UC seamănă cu cea a UE în privinţa realizării şi se încheie printr-o etapă de programare
- realizarea microprogramată a UC a unui sistem numeric, care realizează un algoritm dat, este caracterizată de un set de microinstrucţiuni:
  - microinstrucţiuni de test şi comandă
  - microinstrucţiuni de test
  - microinstrucţiuni de comandă
  - microinstrucţiuni de apel de subprogram
  - microinstrucţiuni de return dintr-un subprogram etc.

METODA DE SINTEZĂ

Etapele sintezei UC

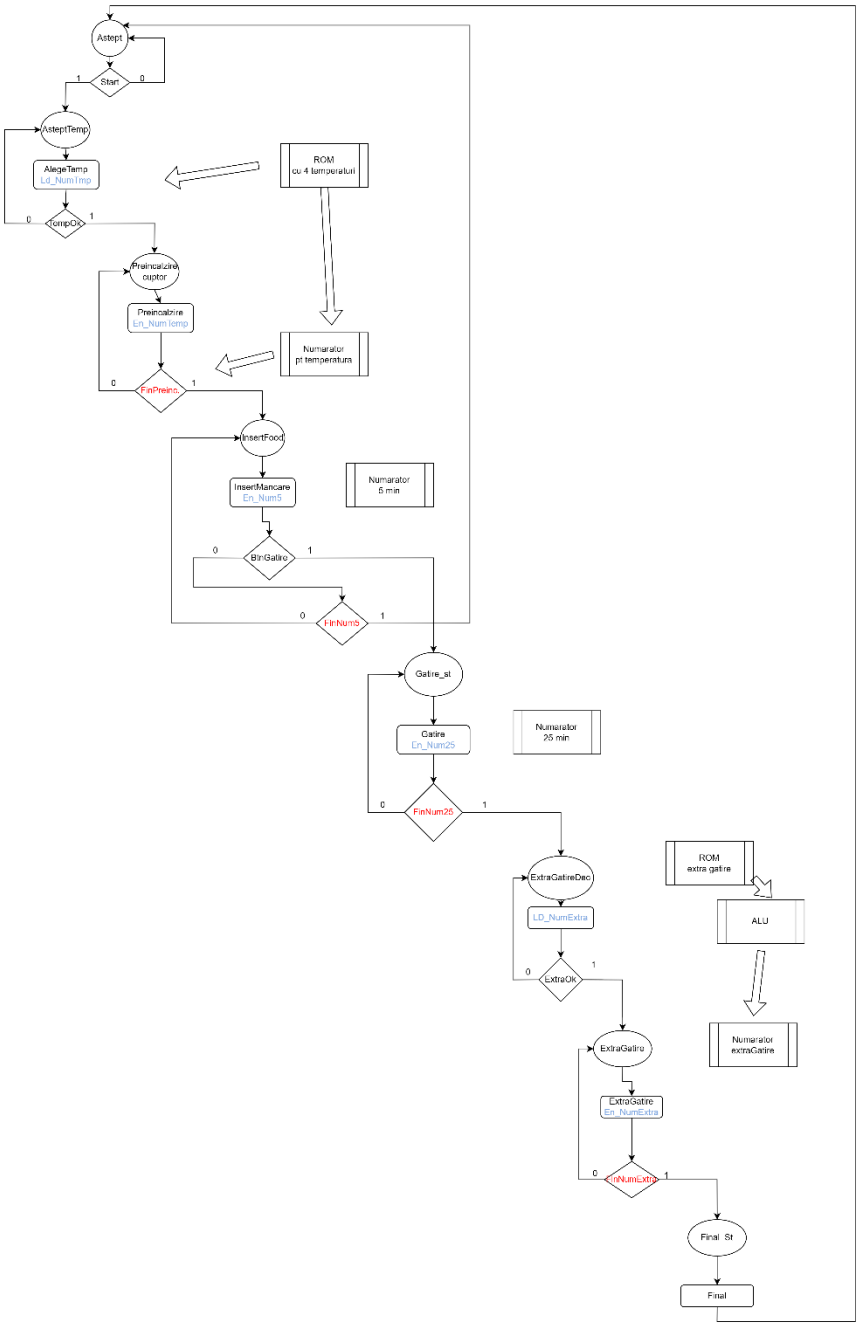
- sinteza se efectuează plecând de la organigramă, prin parcurgerea următoarelor etape:
  - 1. adaptarea eventuală a organigramei la setul de microinstrucţiuni alese pentru UC
  - 2. definirea variabilelor de comandă ale UE
  - 3. redactarea programului initial şi determinarea formatului şi a câmpurilor microinstrucţiunilor
  - 4. declararea registrelor şi resurselor UC şi descrierea funcţională a acesteia cu ajutorul unei organigrame
  - 5. construirea schemei UC şi declararea eventuală a registrelor şi resurselor adiţionale
  - 6. realizarea UC cu ajutorul unor componente combinaţionale şi secvenţiale disponibile/implementate
  - 7. adaptarea programului initial şi programarea memoriei

UC CU 2 INSTRUCŢIUNI, CU REGISTRU DE ADRESĂ

Obiective

- Determinarea setului de microinstrucţiuni
- Determinarea formatului şi câmpurilor microinstrucţiunilor
- Specificarea asocierii cu elementele organigramei

Organigrama UE



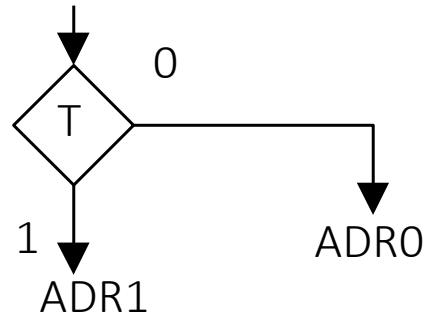
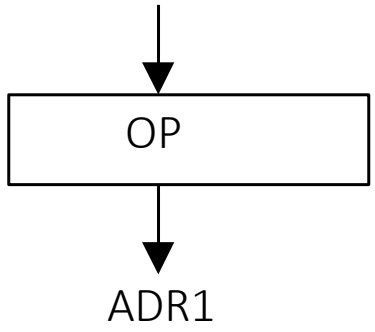
SETUL DE MICROINSTRUCŢIUNI

Implementarea propusa

- Vom asocia:
  - o microinstrucţiune de test binar fiecărui romb (test)
  - o microinstrucţiune de comandă fiecărui cerc (operaţie/stare)
- se aplică direct organigramei originare a sistemului numeric care realizează un cuptor simplificat

SETUL DE MICROINSTRUCŢIUNI

Implementarea propusa

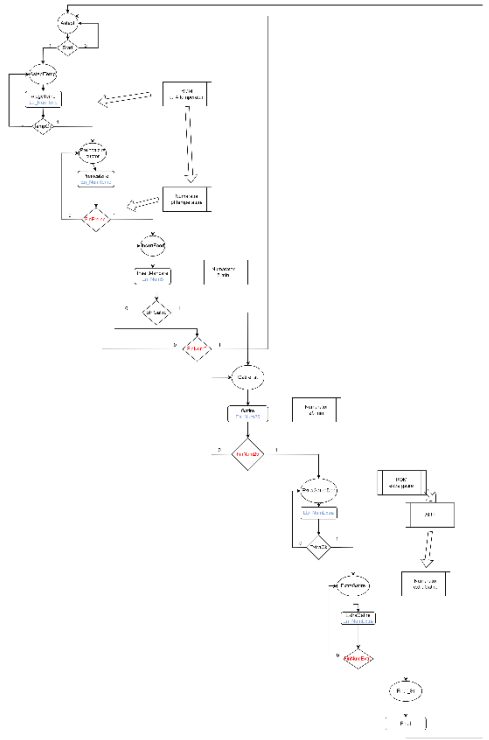
IF T = 0 THEN GOTO ADR0 ELSE GOTO ADR1	ADR 
DO OP AND GOTO ADR1	ADR 

VARIABILE DE COMANDĂ A UE

- fiecare dreptunghi al organigramei originare conţine o operaţie a UE
- ţinând cont de resursele şi de registrele alese pentru realizarea UE, se definesc variabilele sale de comandă, pentru ca UE să execute ansamblul operaţiilor organigramei
- valorile care trebuie să se atribuie variabilelor sunt precizate în tabela operaţiilor UE (rezultă din tabelele operaţiilor proprii pentru resursele şi registrele utilizate la proiectarea UE)

VARIABILE DE COMANDĂ A UE

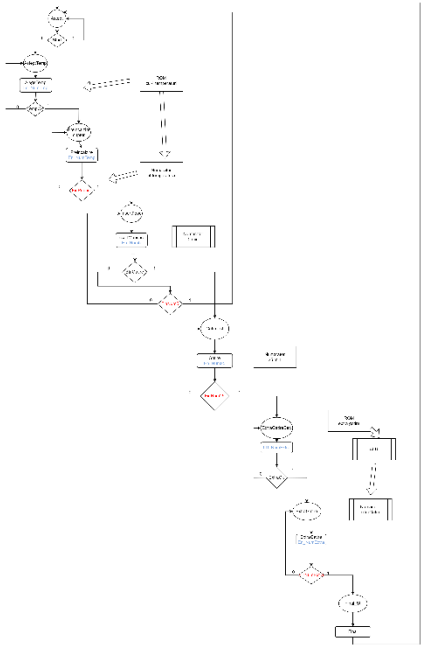
Operaţiile organigramei UE



Operaţie	Descriere	Operatii
OP0	Astept	NOP
OP1	AsteptTemp	CntTmp <- ROM_temp[OptiuneTemp]
OP2	Preincalzire cuptor	CntTmp <- CntTmp – 1
OP3	InsertFood	Cnt5 <- Cnt5 + 1
OP4	Gatire_st	Cnt25 <- Cnt25 + 1
OP5	Extra_gatire_Dec	Cntextra <- 50 ± ROMextra[Gatire_Extra]
OP6	Extra Gatire	Cntextra <- Cntextra - 1
OP7	Final	All signals <- 0
OP8	Test	NOP

VARIABILE DE COMANDĂ A UE

Op	EnTmp	LdTmp	En25	En5	LdExtra	EnExtra	Rst
OP0	0	0	0	0	0	0	0
OP1	0	1	0	0	∅	0	0
OP2	1	0	0	0	0	0	0
OP3	0	0	0	1	0	0	0
OP4	0	0	1	0	0	0	0
OP5	0	∅	0	0	1	0	0
OP6	0	0	0	0	0	1	0
OP7	0	0	0	0	0	0	1
OP8	0	0	0	0	0	0	0



Operaţie	Descriere	Operatii
OP0	Astept	NOP
OP1	AsteptTemp	CntTmp <- ROM_temp[OptiuneTemp]
OP2	Preincalzire cuptor	CntTmp <- CntTmp – 1
OP3	InsertFood	Cnt5 <- Cnt5 + 1
OP4	Gatire_st	Cnt25 <- Cnt25 + 1
OP5	Extra_gatire_Dec	Cntextra <- 50 ± ROMextra[Gatire_Extra]
OP6	Extra Gatire	Cntextra <- Cntextra - 1
OP7	Final	All signals <- 0
OP8	Test	NOP

VARIABILE DE COMANDĂ A UE

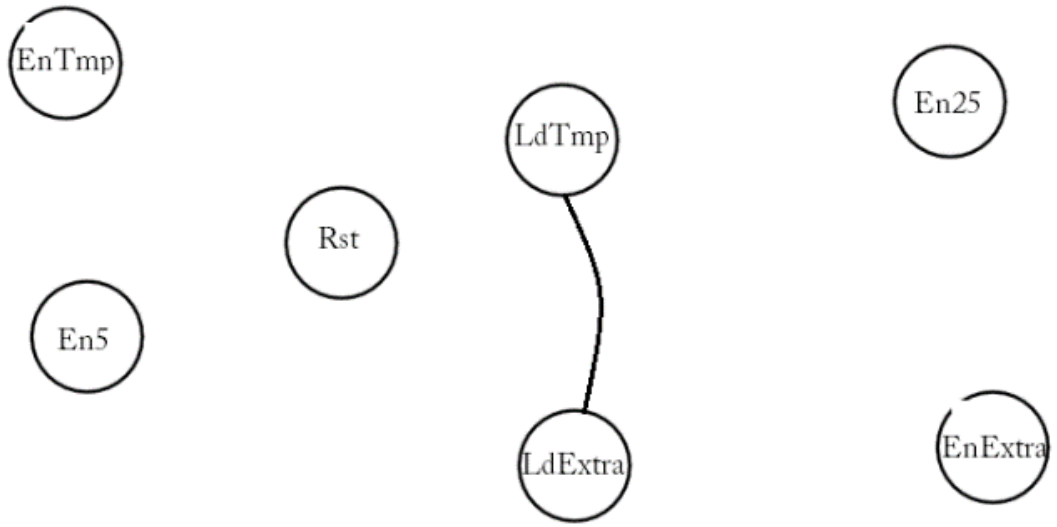
Observaţii

- fiecare romb al organigramei corespunde unei instrucţiuni de test
  - UE trebuie să rămână inactivă în cursul execuţiei unui test $\Rightarrow$ UE efectuează atunci o instrucţiune neutră NOP
  - pentru a simplifica mai mult concepţia UC, admitem în plus că toate variabilele sale de ieşire (adică toate variabilele de comandă a UE) sunt egale cu 0 în timpul unei instrucţiuni de test
  - tabela operaţiilor UE conţine în mod obligatoriu operaţia NOP

VARIABILE DE COMANDĂ A UE

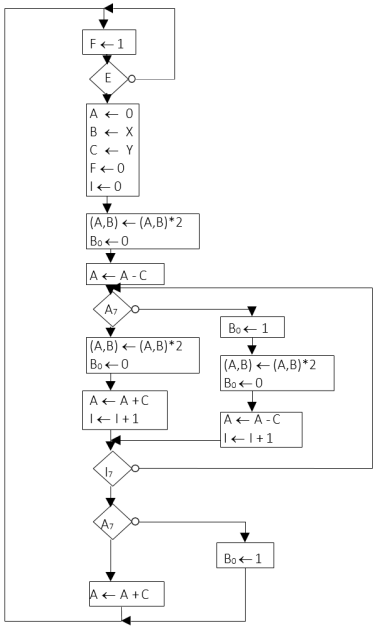
Ansamblul minimal de variabile de comandă pentru UE

- determinarea ansamblului minimal al variabilelor distincte CMD, care trebuie să fie generate de UC pentru UE se face cu ajutorul unui graf de compatibilităţi



VARIABILE DE COMANDĂ A UE (ex. 2)

Op	A_A	SH_A	LD_A	G_B	S_B	SH_B	LD_B	WS_C	J_F	K_F	CLR_I	LD_I	P_I	A_{AU}
OP0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
OP1	∅	0	0	0	∅	0	0	0	1	0	0	0	0	∅
OP2	0	0	1	0	0	0	1	1	0	1	1	∅	∅	∅
OP3	∅	1	0	∅	∅	1	∅	0	0	0	0	0	0	∅
OP4	∅	0	0	0	1	0	1	0	0	0	0	0	0	∅
OP5	1	0	1	0	∅	0	0	0	0	0	0	0	0	0
OP6	1	0	1	0	∅	0	0	0	0	0	0	0	0	1
OP7	1	0	1	0	∅	0	0	0	0	0	0	0	1	0
OP8	1	0	1	0	∅	0	0	0	0	0	0	0	1	1

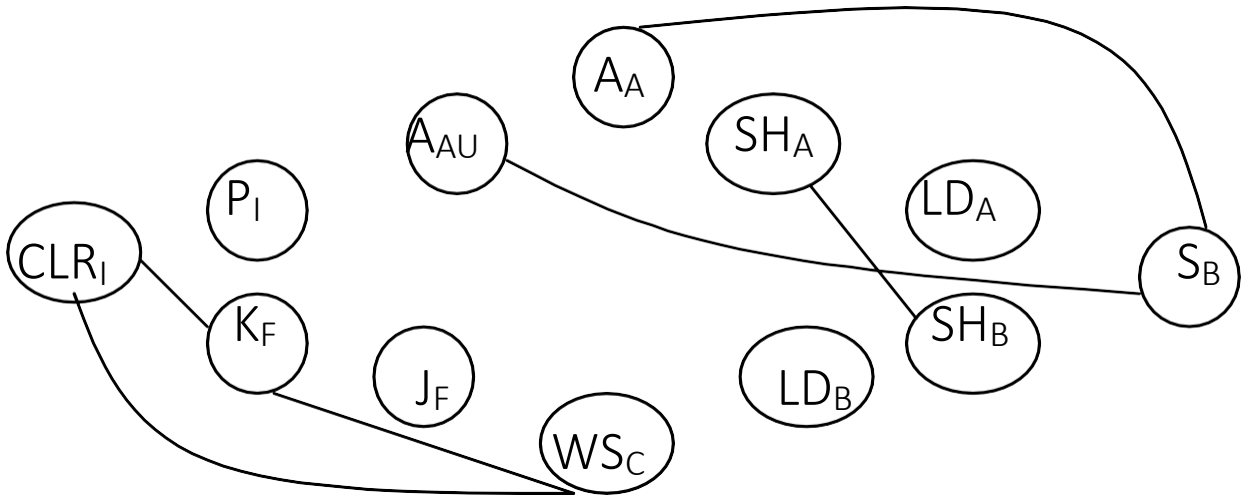


Operaţie	Descriere
OP0	NOP
OP1	$F \leftarrow 1$
OP2	$A \leftarrow 0$; $B \leftarrow X$; $C \leftarrow Y$; $F \leftarrow 0$; $I \leftarrow 0$
OP3	$(A,B) \leftarrow (A,B) * 2$; $B_0 \leftarrow 0$
OP4	$B_0 \leftarrow 1$
OP5	$A \leftarrow A + C$
OP6	$A \leftarrow A - C$
OP7	$A \leftarrow A + C$; $I \leftarrow I + 1$
OP8	$A \leftarrow A - C$; $I \leftarrow I + 1$

VARIABILE DE COMANDĂ A UE

Ansamblul minimal de variabile de comandă pentru UE

- determinarea ansamblului minimal al variabilelor distincte CMD, care trebuie să fie generate de UC pentru UE se face cu ajutorul unui graf de compatibilităţi



VARIABILE DE COMANDĂ A UE

Ansamblul minimal de variabile de comandă pentru UE

- se procedează astfel:
  - **1.** căutăm variabilele de comandă care pot rămâne constante. Atribuirea unor valori particulare condiţiilor indiferente din tabela operaţiilor UE reduce 2 variabile la starea de constante: $G_B = 0$ si $LD_I = 0$ (adică $LD_I = 1$)
  - **2.** fiecare variabilă care rămâne va constitui un nod în graf
  - **3.** compatibilitatea a 2 variabile (faptul că au valori egale pentru fiecare operaţie în care sunt ambele specificate) e indicată printr-un arc neorientat care uneşte cele 2 noduri

VARIABILE DE COMANDĂ A UE

Ansamblul minimal de variabile de comandă pentru UE

- se procedează astfel:
  - **4.** un poligon complet = un ansamblu de noduri care sunt toate conectate 2 câte 2
  - **5.** ansamblul minimal al poligoanelor complete corespunde ansamblului minimal al variabilelor distincte, notat CMD
- din graful de compatibilităţi se pot obţine 2 ansambluri minimale de poligoane complete
- se determină 2 ansambluri minimale de 8 variabile distincte

VARIABILE DE COMANDĂ A UE

Ansamblul minimal de variabile de comandă pentru UE - 2 variante

Varianta 1

- CMD0 = A_{AU}
- CMD1 = P_I
- CMD2 = J_F
- CMD3 = $WS_C = K_F = CLR_I$
- CMD4 = LD_B
- CMD5 = LD_A
- CMD6 = $SH_A = SH_B$
- CMD7 = $A_A = S_B$

Varianta 2

CMD0 = $S_B = A_{AU}$

CMD1 = P_I

CMD2 = J_F

CMD3 = $WS_C = K_F = CLR_I$

CMD4 = LD_B

CMD5 = LD_A

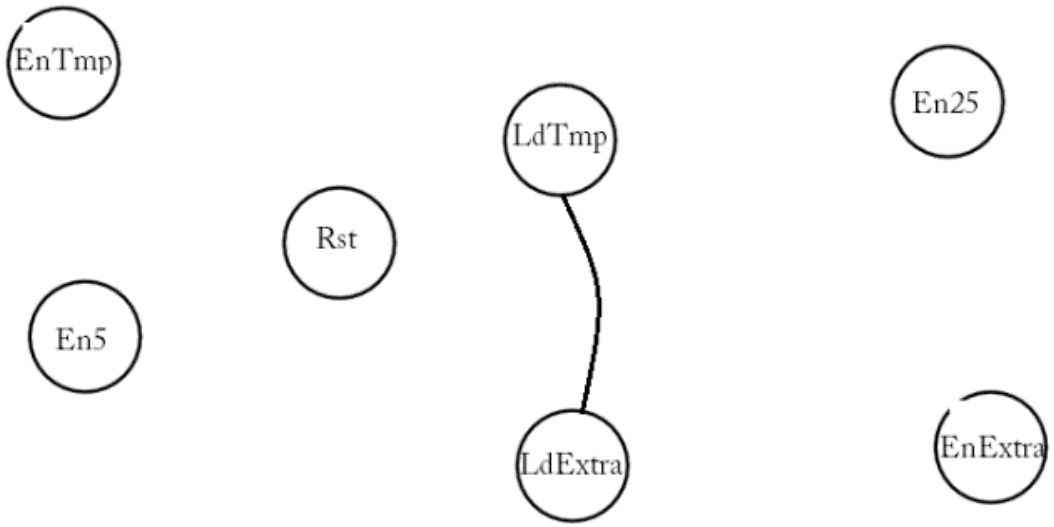
CMD6 = $SH_A = SH_B$

CMD7 = A_A

VARIABILE DE COMANDĂ A UE

Ansamblul minimal de variabile de comandă pentru UE – Exemplu cuptor

- CMD0 = EnTmp
- CMD1 = LdTmp = LdExtra
- CMD2 = EnExtra
- CMD3 = En5
- CMD4 = En25
- CMD5 = Rst



VARIABILE DE COMANDĂ A UE

Ansamblul minimal de variabile de comandă pentru UE

- în tabel apar valorile binare pentru variabilele de comandă a UE, corespunzătoare cazului când toate condiţiile indiferente sunt alese 0
- Afisam valorile si in hexa

Operaţie	CMD5...0	CMD5...0
OP0	000000	x00
OP1	000010	x02
OP2	000001	x01
OP3	001000	x08
OP4	010000	x10
OP5	000010	x02
OP6	000100	x04
OP7	100000	x20
OP8	000000	x00

PROGRAMUL INITIAL

Programul initial pt UC

- redactarea se face transcriind organigrama initiala a UE cu ajutorul setului de microinstrucţiuni
- programul este cel ce va fi înscris în memoria ROM
- fiecare din etichetele NEXT specifică adresa microinstrucţiunii care ocupă linia următoare din program
- pentru salturi se utilizează etichete

PROGRAMUL PT UC

0) Astept: IF Start=1 THEN GOTO AsteptTemp
ELSE GOTO Astept

1) AsteptTemp: DO OP1 AND GOTO NEXT

2) Test_Pre: IF TempOk=1 THEN GOTO Preincalzire_St
ELSE GOTO AsteptTemp

3) Preincalzire_St: DO OP2 AND GOTO NEXT

4) Test_FinPre: IF FinCntTmp=1 THEN GOTO InsertFood
ELSE GOTO PreIncalizre_St

5) InsertFood: DO OP3 AND GOTO NEXT

6) Test_BtnGatire: IF BtnGatire=1 THEN GOTO Gatire_St
ELSE GOTO Next

7) Test_FinCnt5: IF FinCnt5=1 THEN GOTO Astept
ELSE GOTO InsertFood

8) Gatire_St: DO OP4 AND GOTO NEXT

9) Test_FinCnt25: IF FinCnt25=1 THEN GOTO ExtraGatireDec
ELSE GOTO Gatire_St

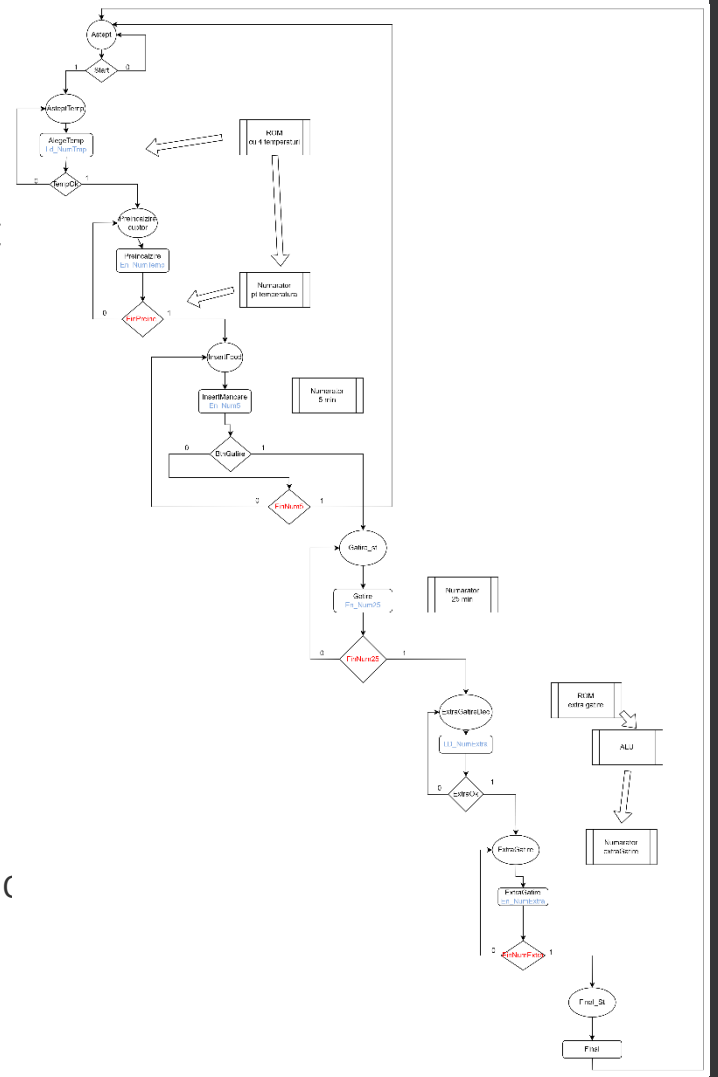
10) ExtraGatireDec: DO OP5 AND GOTO NEXT

11) Test_ExtraOk IF ExtraOk=1 THEN GOTO ExtraGatire
ELSE GOTO ExtraGatireDec

12) ExtraGatire DO OP6 AND GOTO NEXT

13) Test_FinCntExtra: IF FinCntExtra=1 THEN GOTO Final_St
ELSE GOTO ExtraGatire

14) Final_St: DO OP7 AND GOTO Astept



DEFINIREA MICROINSTRUCŢIUNILOR

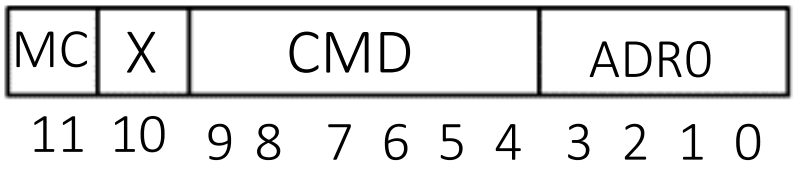
Formatul şi câmpurile microinstrucţiunilor

- determinate de:
  - numărul microinstrucţiunilor din set
  - dimensiunea programului initial
  - ansamblul minimal de variabile de comandă
  - numărul variabilelor de test ale organigramei

DEFINIREA MICROINSTRUCŢIUNILOR

Formatul şi câmpurile microinstrucţiunilor

- microinstrucţiunea de **test** definită de:
  - un bit de cod MC = 0
  - un câmp TEST pentru selecţia variabilei de test conţinută în romb
  - 2 câmpuri ADR0 şi ADR1 care dau, respectiv, adresa microinstrucţiunii următoare, pentru valoarea 0 şi pentru valoarea 1 a variabilei de test

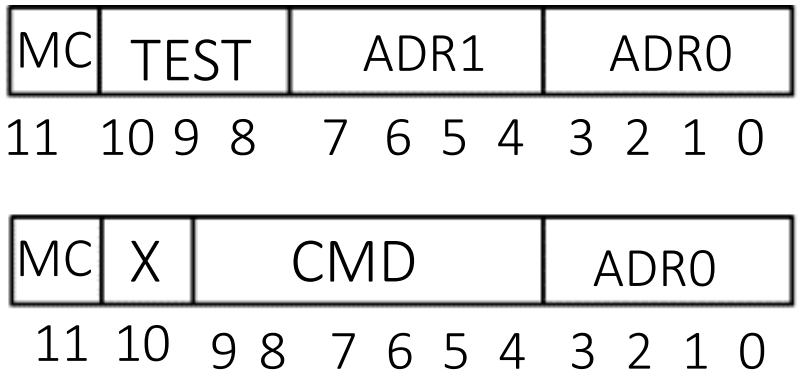


DEFINIREA MICROINSTRUCŢIUNILOR

Formatul şi câmpurile microinstrucţiunilor

- microinstrucţiunea de **execuţie** definită de:
  - un bit de cod MC = 1
  - un câmp CMD/OP care dă starea variabilelor de comandă

  CMD a UE pentru efectuarea operaţiei conţinute în dreptunghi
  - un câmp ADR0 care dă adresa microinstrucţiunii următoare din program



DECLARAREA ŞI DESCRIEREA FUNCŢIONALĂ A UC

Registre şi resurse pentru UC

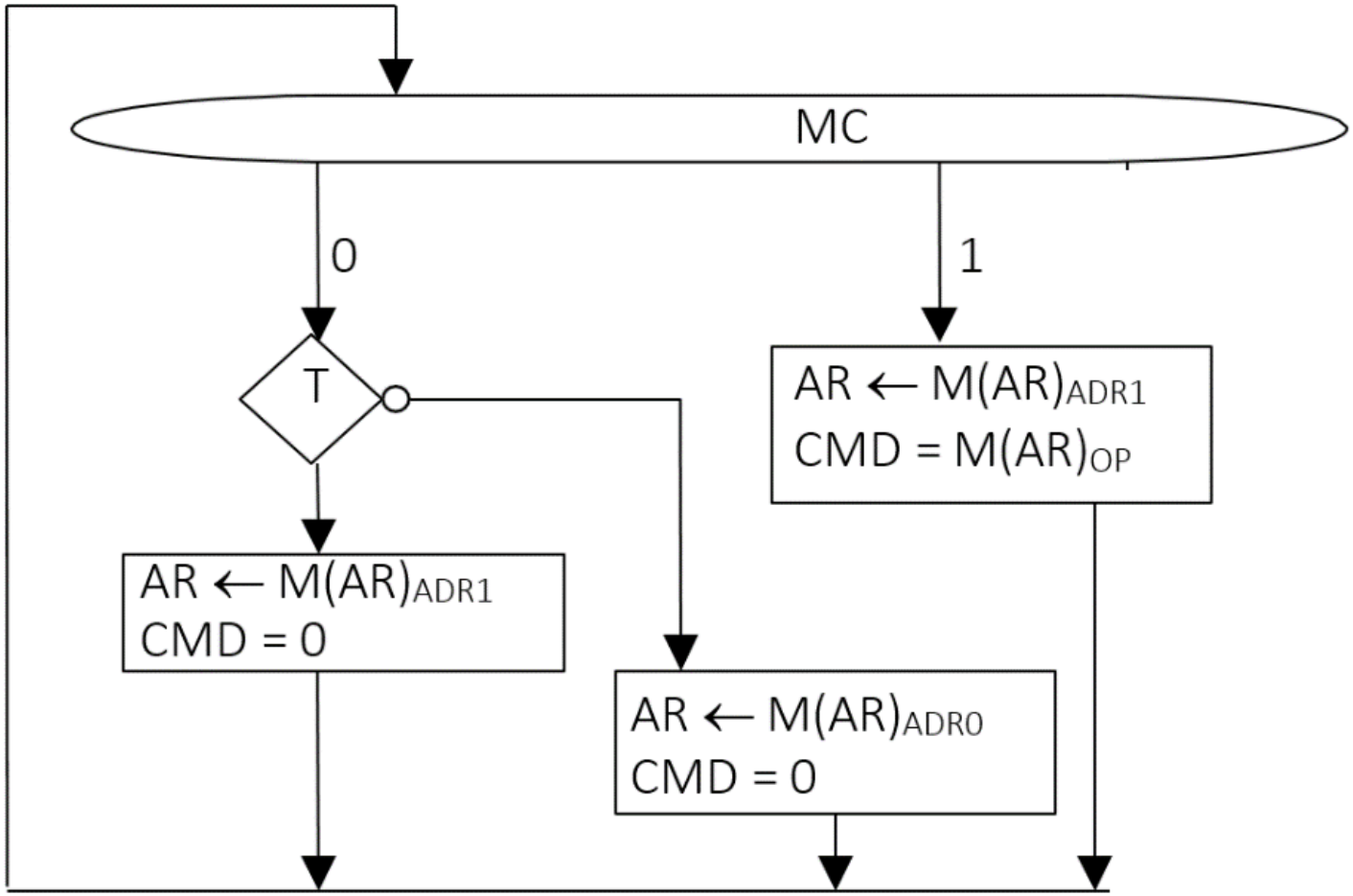
- pentru a executa cele 2 microinstrucţiuni, UC are nevoie de:
  - AR_{1x4} = registru de adresă (Address Register) a memoriei ROM
  - $M_{15\ loc\ x12\ b}$ = memorie de microprogram (4b/adresa)
  - $T_{8:1}$ = multiplexor de test
  - CMD_{1x6} = registru pentru păstrarea variabilelor de comandă a UE

DECLARAREA ŞI DESCRIEREA FUNCŢIONALĂ A UC

Organigrama UC

- organigrama UC descrie funcţionarea pentru fiecare din microinstrucţiuni, în funcţie de codul lor MC şi pentru selectia efectuată in functie de semnalul de intrare (poate fi de la utilizator – start; sau de la UE – FinCnt25)
- $M(AR)_{ADR}$ = conţinutul locaţiei (cuvântului) din memoria M, de la adresa dată de AR, partea ADR din această locaţie

ORGANIGRAMA UC

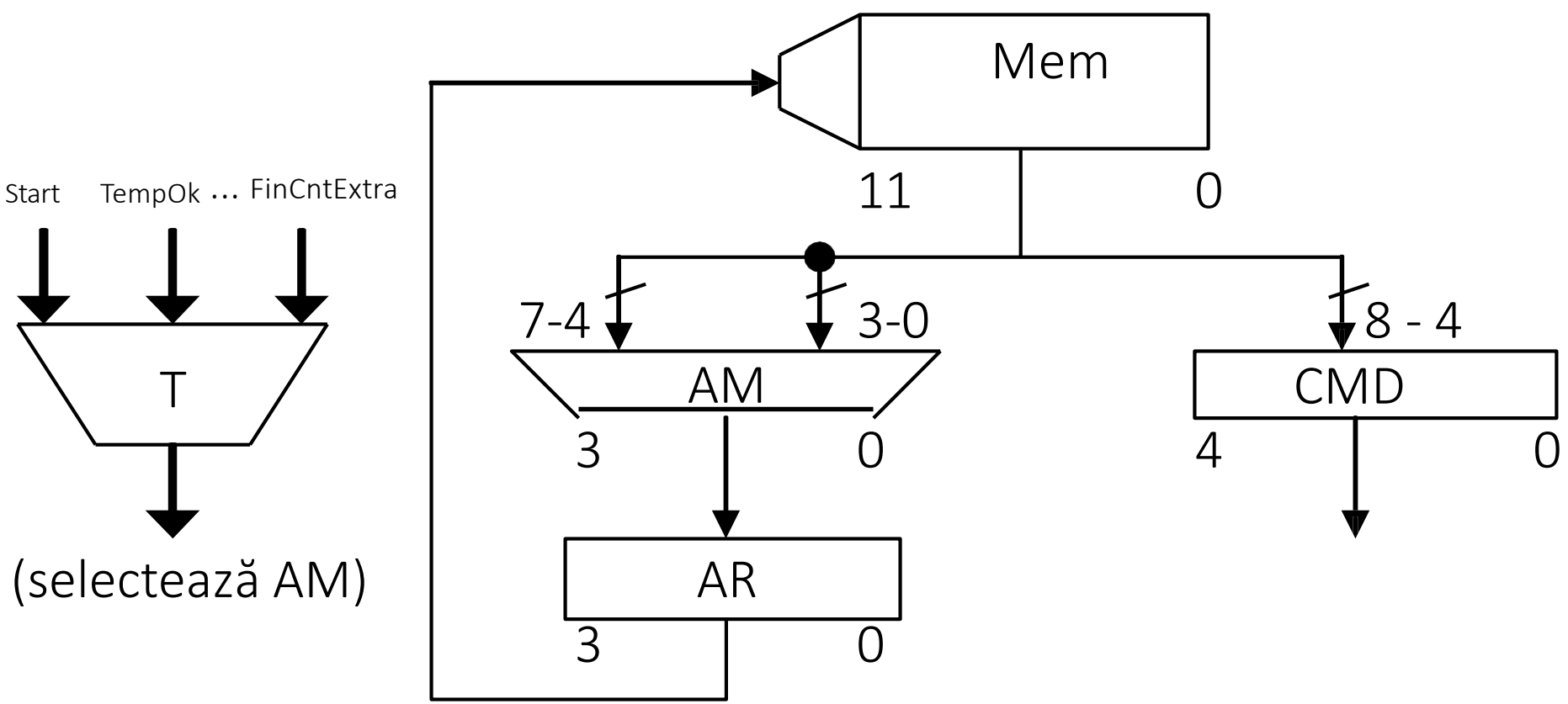


DECLARAŢIE ADIŢIONALĂ

Resurse adiţionale

- schema UC mai are nevoie pe lângă elementele anterioare de:
  - AM_{1x4} = multiplexor de adresă, pentru a selecta una sau cealaltă dintre cele 2 adrese (ADR0 sau ADR1) furnizate de cuvintele din memorie

SCHEMA UC


Start
TempOk
...
FinCntExtra
T
(selectează AM)
Mem
11
0
7-4
3-0
8 - 4
AM
CMD
3
0
4
0
AR
3
0

REALIZAREA UC

Operaţii ale UC

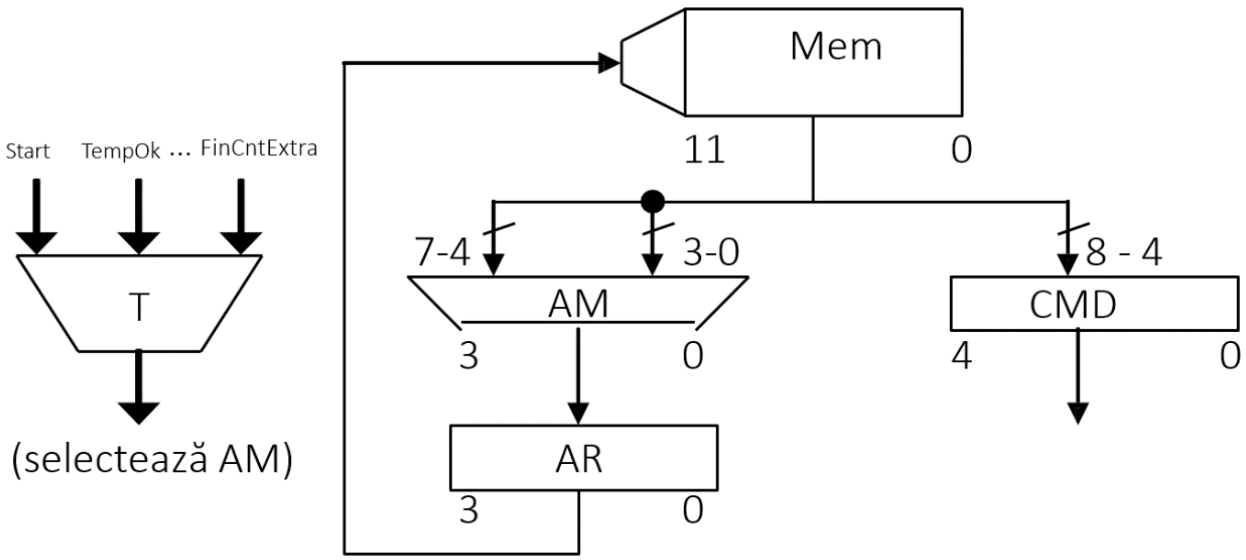
- operaţiile descrise în organigrama UC ţin cont de multiplexorul de adrese AM
- realizarea UC se reduce la alegerea unui ansamblu de registre şi resurse capabile să execute aceste operaţii

Operaţie	Descriere
OP1	$AR \leftarrow AM; AM = M(AR)_{ADR1}; CMD = 0$
OP2	$AR \leftarrow AM; AM = M(AR)_{ADR0}; CMD = 0$
OP3	$AR \leftarrow AM; AM = M(AR)_{ADR0}; CMD = M(AR)_{OP}$

REALIZAREA UC

Componente ale UC

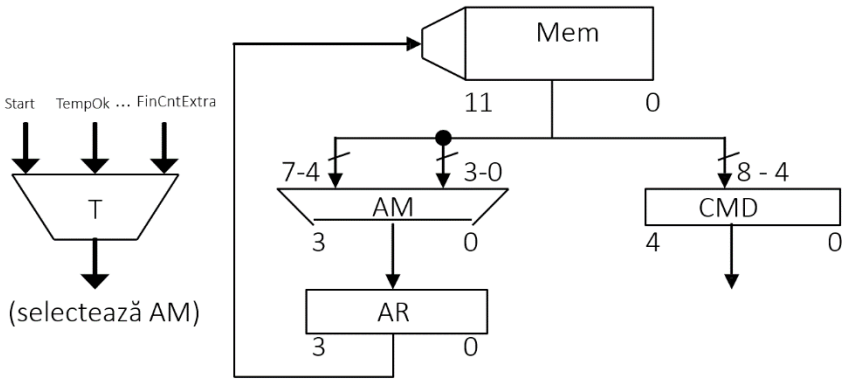
- operaţiile descrise în organigrama UC ţin cont de multiplexorul de adrese AM
- realizarea UC se reduce la alegerea unui ansamblu de registre şi resurse capabile să execute aceste operaţii



REALIZAREA UC

Componente ale UC

- Registru de adrese AR – registru pe 4 biti
- AM - se foloseşte un multiplexor 2:1 pe 4 biţi
- registrul CMD - se foloseste un registru pe 6 biti
- Memoria M – se foloseste o memorie ROM, cu 4 biti de adresa (16 locatii); Datele la fiecare locatie vor fi pe 12 biti
  - Programul de control va fi incarcat in memoria ROM



REALIZAREA UC

Componente ale UC

- Registru de adrese AR – registru pe 4 biti
- AM - se foloseşte un multiplexor 2:1 pe 4 biţi
- registrul CMD - se foloseste un registru pe 6 biti
- Memoria M – se foloseste o memorie ROM, cu 4 biti de adresa (16 locatii); Datele la fiecare locatie vor fi pe 12 biti
  - Programul de control va fi incarcat in memoria ROM

REALIZAREA UC

Multiplexorul de test T

- multiplexorul de test T selectează intrarile corespunzătoare microinstrucţiunilor de test binar din programul UC
- operaţiile apar în detaliu în tabel, care precizează starea variabilelor de comandă ale multiplexorului 8:1 utilizat
- prin codificarea utilizată în tabel se pune în corespondenţă starea câmpului TEST a microinstrucţiunii de test cu 3 dintre variabilele de selecţie, A, B, C ale multiplexorului
- Intrarile de date ale MUX vor fi intrarile UC (fie de la utilizator, fie de la UC)
- Va avea un Enable, pentru a fi folosit doar pt MC=1

REALIZAREA UC

Multiplexorul de test T

- multiplexorul de test T selectează intrarile corespunzătoare microinstrucţiunilor de test binar din programul UC
- operaţiile apar în detaliu în tabel, care precizează starea variabilelor de comandă ale multiplexorului 8:1 utilizat
- prin codificarea utilizată în tabel se pune în corespondenţă starea câmpului TEST a microinstrucţiunii de test cu 3 dintre variabilele de selecţie, A, B, C ale multiplexorului

Operatie	TEST	Stare asociata	Sel0	Sel1
T ← Start	ABC = 000	Astept	Astept	AsteptTemp
T ← TempOk	ABC = 001	Test_FinPre	AsteptTemp	Preincalzire_st
T ← FinCntTemp	ABC = 010	Test_InsFood	Preincalzire_st	InsertFood
T ← BtnGatire	ABC = 011	Test_BtnGatire	Test_FinCnt5	Gatire_St
T ← FinCnt5	ABC = 100	Test_FinCnt5	InsertFood	Astept
T ← FinCnt25	ABC = 101	Test_FinCnt25	Gatire_St	ExtraGatireDec
T ← ExtraOk	ABC = 110	Test_ExtraOk	ExtraGatireDec	ExtraGatire
T ← FinCntExtra	ABC = 111	Test_FinCntExtra	ExtraGatire	Final_St

PROGRAMAREA

- programul în hexazecimal al memoriei M a UC, este scris pentru situaţia în care câmpul nedefinit (∅) al microinstrucţiunii de test binar este ales egal cu 0
- redactarea acestui program se realizează pe baza programului adaptat al sistemului numeric, ţinând cont de:
  - formatul microinstrucţiunilor
  - codificarea variabilelor de test
  - codificarea variabilelor de comandă ale UE
- dimensiunea programului este de 15x12 biţi = 180 biţi

PROGRAMAREA

Adresă (în hexa)	Continut (in binar)	Conţinut (în hexa)
0	b”1_000_0001_0000”	810
1	b”0_0_000010_0010”	022
2	b”1_001_0011_0001”	931
3	b”0_0_000001_0100”	014
4	b”1_010_0101_0011”	A53
5	b”0_0_001000_0110”	086
6	b”1_011_1000_0111”	B87
7	b”1_100_0000_0101”	C05
8	b”0_0_010000_1001”	109
9	b”1_101_1010_1000”	DA8
A	b”0_0_000010_1011”	02B
B	b”1_110_1100_1010”	ECA
C	b”0_0_000100_1101”	04D
D	b”1_111_1110_1100”	FEC
E	b”0_0_100000_0000”	200