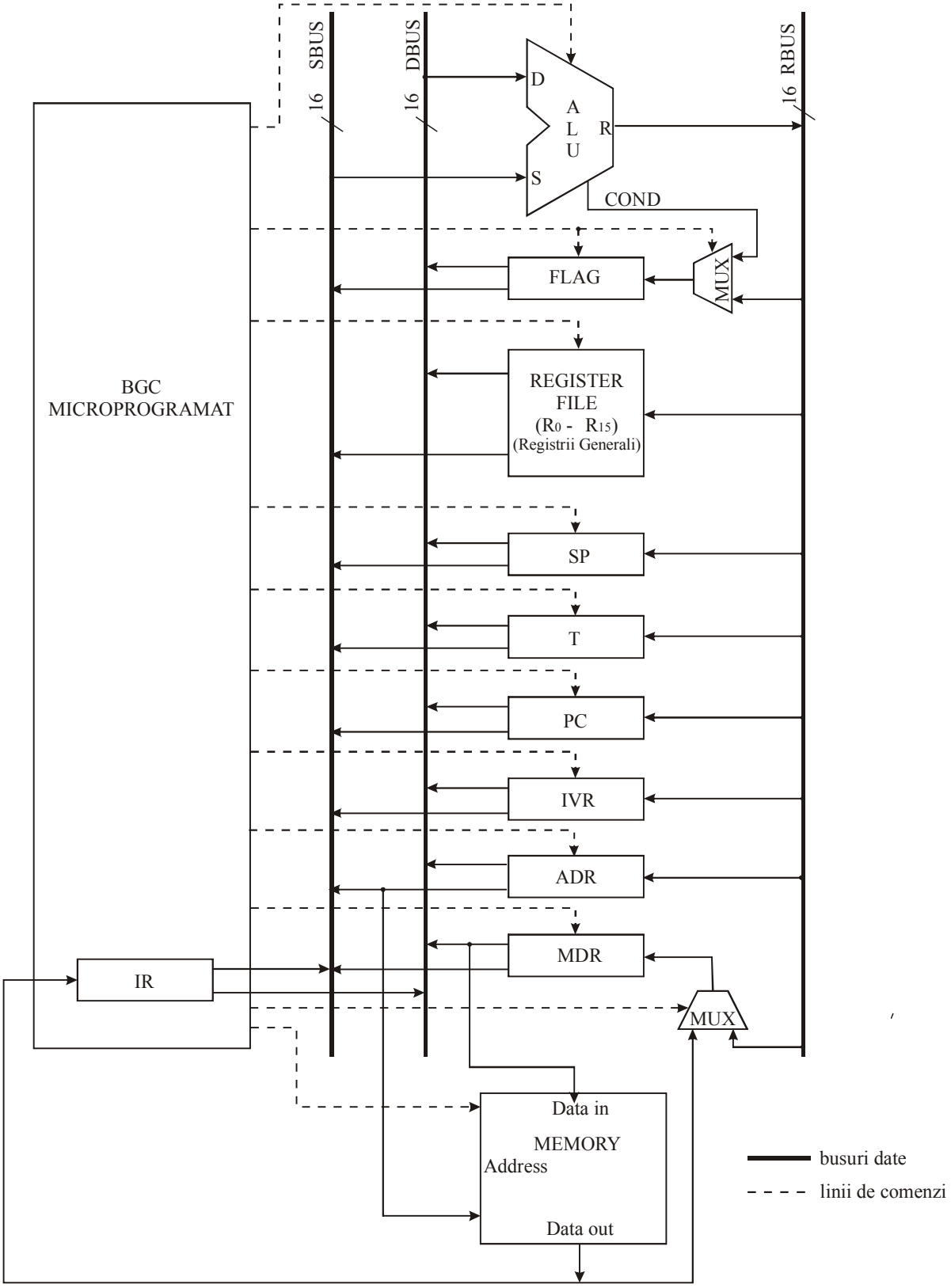
Tema proiect:

PROIECTAREA UNITĂȚII DE CONTROL MICROPROGRAMATE AFERENTĂ UNUI PROCESOR CISC

A. ARHITECTURA PROCESORULUI:



S_{BUS} – busul operandului sursă
D_{BUS} – busul operandului destinație
R_{BUS} – busul rezultatului
FLAG – registrul de *flag*-uri: N, Z, V, C
REGISTER FILE – *file* de registre generale (16 registre×16 biți notate R0÷R15)
SP – *stack pointer* (registrul pointer de stivă)
T – registru tampon (pentru memorări temporare – invizibil programatorului)
PC – *program counter (instruction pointer).*
IVR – *interrupt vector register* (registrul vectorului de întrerupere)
ADR – *address register* (registrul de adrese); are rolul de a adresa locațiile de memorie
MDR – *memory data register* (registru de date aferent memoriei); furnizează datele de scris în memorie în ciclurile de scriere şi respectiv este încărcat cu datele citite din memorie în ciclurile de citire
IR – *instruction register* (registrul instrucțiunii)

B. FORMATUL INSTRUCȚIUNII

Setul de instrucțiuni este format din patru clase de instrucțiuni:

b1) instrucțiuni cu doi operanzi

Cei doi operanzi sunt denumiți operand sursă şi respectiv operand destinație. Pentru localizarea lor se vor defini patru moduri de adresare: imediat, registru direct, registru indirect şi indexat. În funcție de modul de adresare operandul (sursă sau destinație) se poate afla într-un registru general sau într-o locație de memorie. Setul de instrucțiuni este perfect ortogonal, permițând orice combinație în ceea ce priveşte localizarea celor doi operanzi:

Localizare operand sursă	Localizare operand destinație
registru	registru
registru	memorie
memorie	registru
memorie	memorie

Formatul instrucțiunii cu doi operanzi este:

4biti	2biti	4biti	2biti	4biti
OPCODE	MAS	RS	MAD	RD

OPCODE – *opcode*-ul instrucțiunii (codul operației)
MAS/MAD – mod adresare (operand) sursă/destinație
RS/RD – registrul (general utilizat pentru adresarea operandului) sursă/destinație

În cazul acestor instrucțiuni rezultatul se depune peste operandul destinație care se va pierde.
În această clasă se definesc următoarele instrucțiuni:

- instrucțiuni de transfer:

```
MOV dest, src      ;op. dest. ← op. src
                   ;op. dest.=operand destinație
                   ;op. src.=operand sursă
```

Exemple:

```
MOV R0,R1          ;R0 ← R1
MOV R4,(R2)        ;R4 ← locația de memorie adresată de R2
```

```
MOV (R3), 124(R5) ;locația de memorie adresată de R3 ← locația de memorie
                  ;adresată de R5+124; 124 va fi indexul (adresare indexată)
```

- instrucțiuni aritmetice:

```
ADD dest, src     ; op. dest. ← op. dest. + op. src.
SUB dest, src     ; op. dest. ← op. dest. - op. src.
```

Exemple:

```
ADD (R6),R0       ; locația de memorie adresată de R6 ← locația de memorie
                  ; adresată de R6 + R0
SUB R3,R5         ; R3 ← R3 – R5
```

- instrucțiuni logice:

```
CMP dest, src     ; op. dest. – op. src. (fără depunerea rezultatului ci doar
                  ; pentru poziționarea flag-urilor de condiții)
AND dest, src     ; op.dest. ← op.dest AND op.src.
OR dest, src      ; op.dest. ← op.dest OR op.src.
XOR dest, src     ; op.dest. ← op.dest XOR op.src.
```

Exemple:

```
CMP R0,(R1)       ; R0 – locația de memorie adresată de R1 cu poziționarea
                  ; flag-urilor de condiții conform cu rezultatul scăderii
AND R2,R4         ; R2 ← R2 AND R4
OR R1,(R5)        ; R1 ← R1 OR locația de memorie adresată de R5
XOR R3,R0         ; R3 ← R3 XOR R0
```

b2) instrucţiuni cu un operand

Operandul unic referit de instrucțiunile din această clasă va fi operandul destinație care va fi localizat pe baza aceloraşi patru moduri de adresare.

Formatul instrucțiunilor cu un operand este:

10biti	2biti	4biti
OPCODE	MAD	RD

OPCODE – *opcode*-ul instrucțiunii (codul operației)
MAD – mod de adresare operand destinație
RD – registru general utilizat pentru adresarea operandului destinație

În această clasă se definesc următoarele instrucțiuni:

- instrucțiuni logice şi aritmetice:

```
CLR dest          ; dest ← 0
NEG dest          ; dest ← dest (negat)
INC dest          ; dest ← dest+1
DEC dest          ; dest ← dest – 1
```

Exemple:

```
CLR (R0)          ; locația de memorie adresată de R0 ← 0
NEG R3            ; R3 ← R3 (negat)
INC (R2)          ; locația de memorie adresată de R2 ← locația de
                  ; memorie adresată de R2 + 1
DEC R5            ; R5 ← R5 – 1
```

- instrucțiuni de deplasare şi rotire aritmetică şi logică:

```
ASL dest        ; deplasare aritmetică la stânga dest (Arithmetic
                ; Shift Left)
ASR dest        ; deplasare aritmetică la dreapta dest (Arithmetic
                ; Shift Right)
LSR dest        ; deplasare logică la dreapta dest (Logical
                ; Shift Right)
ROL dest        ; rotire la stânga dest (ROtate Left)
ROR dest        ; rotire la dreapta dest (ROtate Right)
RLC dest        ; rotire la stânga cu carry (Rotate Left through
                ; Carry)
RRC dest        ; rotire la dreapta cu carry (Rotate Right through
                ; Carry)
```

Exemple:

```
ASL R1          ; R1 ← R1 deplasat aritmetic la stânga cu o poziție
ASR (R2)        ; locația de memorie adresată de R2 ← conținutul
                ; locației de memorie adresată de R2 deplasat la
                ; dreapta cu o poziție
LSR 14(R0)      ; locația de memorie de la adresa R0+14 ←
                ; conținutul locației de memorie de la adresa
                ; R0+14 deplasat logic la dreapta cu o poziție
RLC R7          ; R7 ← R7 rotit la stânga împreună cu carry
```

- instrucțiunile JMP, CALL, PUSH şi POP

```
JMP adr         ; salt la adresa specificată (adr = operand sursă
                  specificat în instrucțiune)
CALL adr        ; apel procedură specificată de la adresa specificată
                ; (adr = operand sursă specificat în instrucțiune)
PUSH Ri         ; salvare în stivă registru general Ri
POP Ri          ; restaurare din stivă registru general Ri
```

Exemple:

```
JMP 36(R1)      ; salt la adresa R1+36
CALL 1248H      ; apel procedură de la adresa 1248H
PUSH R3         ; stiva ← R3
POP R5          ; R5 ← stiva (conținutul locației din vârful stivei)
```

b3) instrucțiunile de salt (salturi relative la PC denumite aici instrucțiuni de *branch*)

Formatul instrucțiunilor de salt este:

8biti	8biti
OPCODE	OFFSET

OPCODE – *opcode*-ul instrucțiunii (codul operației)

OFFSET – număr cu semn (cod complementar) care indică sensul saltului şi numărul de adrese sărit. Dacă semnul OFFSET-ului este minus, saltul va fi înapoi, iar dacă semnul este plus saltul va fi înainte. Saltul este relativ la PC-ul curent. Prin PC_curent se înțelege adresa instrucțiunii care succede în program instrucțiunii de *branch.*

În această clasă se definesc următoarele instrucțiuni:

```
BR              ; salt relativ necondiționat (branch)
BNE             ; salt dacă flag-ul Z=0 (branch if not equal)
BEQ             ; salt dacă flag-ul Z=1 (branch if equal)
```

```
BPL      ; salt dacă flag-ul S=0 (branch if plus)
BMI      ; salt dacă flag-ul S=1 (branch if minus)
BCS      ; salt dacă flag-ul C=1 (branch if carry is set)
BCC      ; salt dacă flag-ul C=0 (branch if carry is clear)
BVS      ; salt dacă flag-ul V=1 (branch if overflow is set)
BVC      ; salt dacă flag-ul V=0 (branch if overflow is clear)
```

Exemple:

```
BR adr   ; salt necondiționat la adresa adr. Asamblorul va
         ; calcula OFFSET-ul ca rezultat al diferenței
         ; adr – PC_curent)
BEQ ET1  ; salt dacă Z=0 la eticheta ET1 (OFFSET-ul va fi
         ; dat de diferența dintre adresa etichetei specificate
         ; şi PC-ul curent)
```

b4) instrucțiuni diverse

Formatul acestor instrucțiuni este:

16biti

OPCODE

OPCODE – *opcode*-ul instrucțiunii (codul operației)

În această clasă se definesc următoarele instrucțiuni:

- instrucțiuni de poziționare a *flag*-urilor de condiții

```
CLC      ; C ← 0 (clear carry)
CLV      ; V ← 0 (clear overflow)
CLZ      ; Z ← 0 (clear zero)
CLS      ; S ← 0 (clear sign)
CCC      ; C/V/Z/S ← 0 (clear condition code)
SEC      ; C ← 1 (set carry)
SEV      ; V ← 1 (set overflow)
SEZ      ; Z ← 1 (set zero)
SES      ; S ← 1 (set sign)
SCC      ; C/V/Z/S ← 1 (set condition code)
```

- instrucțiunile: NOP, RET, RETI, HALT, WAIT, PUSH PC, POP PC, PUSH FLAG, POP FLAG

```
NOP      ; nici o operație (No OPeration)
RET      ; revenire din procedură (RETurn)
RETI     ; revenire din întrerupere (RETurn from Interrupt)
HALT     ; oprire
WAIT     ; aşteptare (deblocare din WAIT doar prin
         ; întrerupere)
PUSH PC  ; stiva ← PC
POP PC   ; PC ← conținutul locației din vârful stivei
PUSH FLAG ; stiva ← FLAG (registrul de flag-uri)
POP FLAG ; FLAG ← conținutul locației din vârful stivei
```

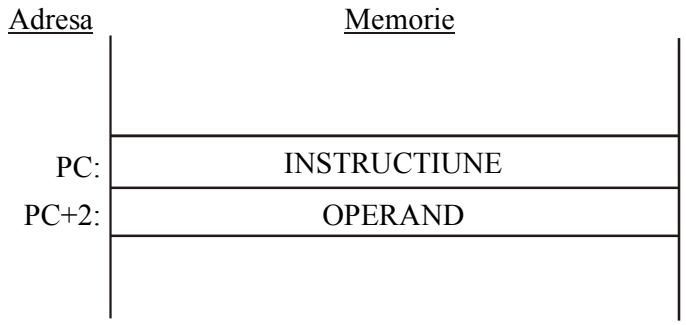
C. Modurile de adresare

MAS/MAD sunt câmpuri de doi biți care permit codificarea a patru moduri de adresare.

MAS/MAD		Denumire mod adresare
0	0	imediat
0	1	registru direct
1	0	registru indirect
1	1	indexat

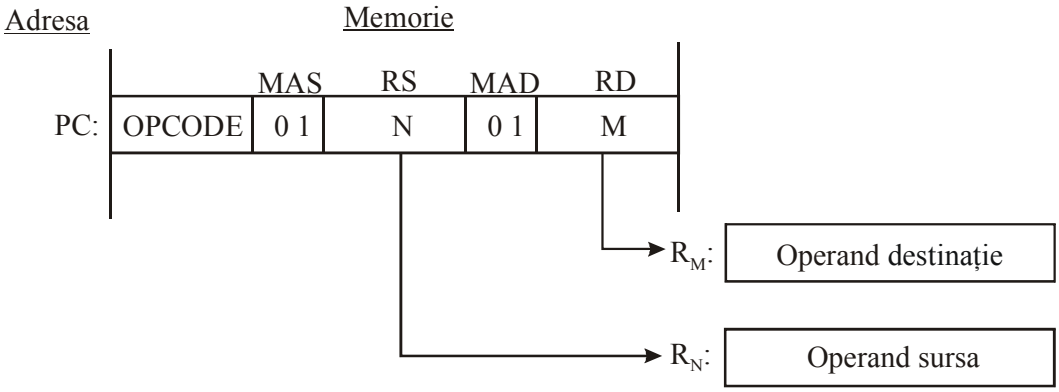
Localizarea operandului în cadrul celor patru moduri de adresare se realizează conform următoarelor scheme de principiu:

c1) imediat:



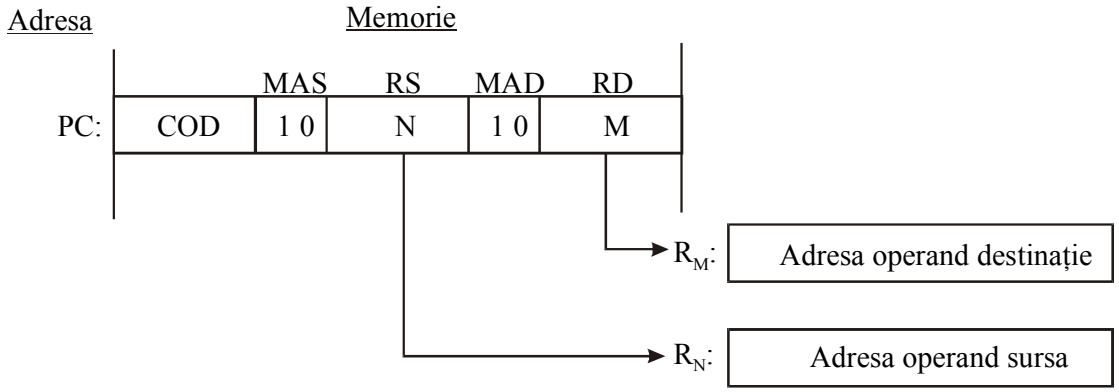
Unitatea adresabilă în memorie este octetul, iar codul unei instrucțiuni este pe 16 biți (1 *word*). Dacă instrucțiunea se află în memorie la adresa PC, atunci operandul imediat se va afla la adresa PC+2 (imediat după instrucțiune). La adresa PC+4 se va găsi codul următoarei instrucțiuni.

c2) registru direct:



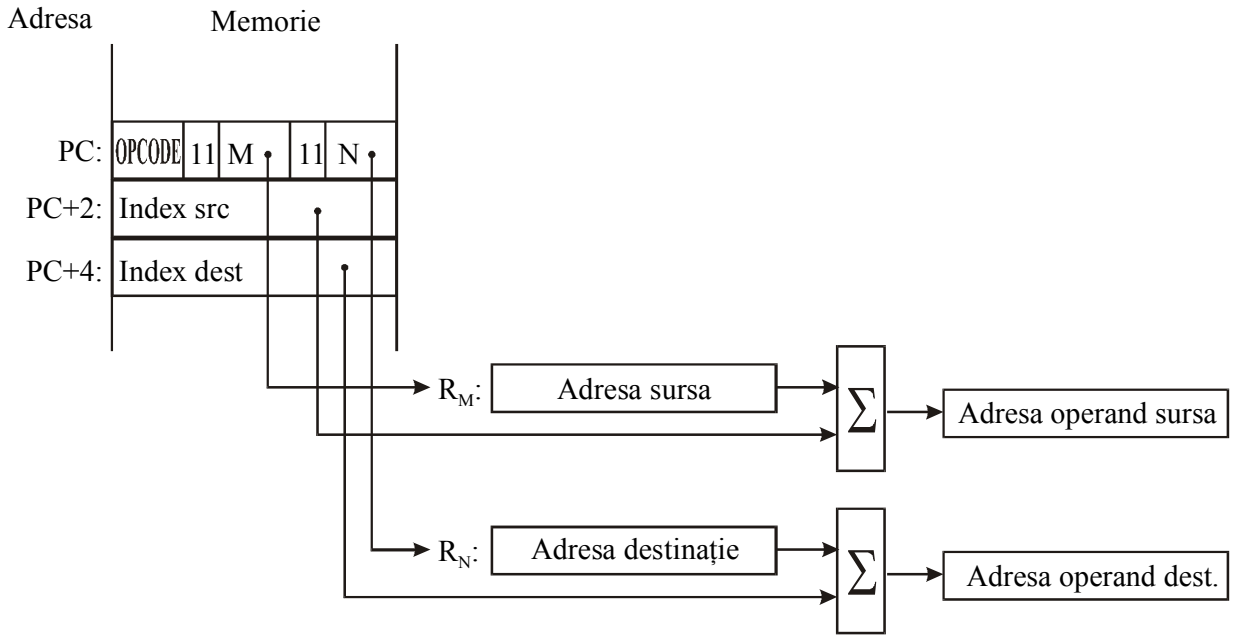
În cazul modului de adresare registru direct operandul sursă (destinație) sa găsește în registrul general selectat de adresa codificată binar în câmpul RS (RD) din codul instrucțiunii. În mod uzual, registrul general selectat de câmpul RS se numește **registru sursă** iar registrul selectat de câmpul RD se numește **registru destinație**. Putem astfel conchide: în cazul modului de adresare registru direct operandul sursă (destinație) sa găsește în registrul sursă (registrul destinație).

c3) registru indirect:



În cazul modului de adresare registru indirect operandul sursă (destinație) se găseşte în memorie. Adresa locației de memorie ce conține operandul sursă (destinație) se găsește în registrul general selectat de adresa codificată binar în câmpul RS (RD) din codul instrucțiunii. În concluzie, în registrul sursă (destinație) nu găsim operandul (ca la c2) ci adresa operandului sursă (destinație).

c4) indexat:



Operandul sursă (destinație) se află în memorie. Adresa operandului sursă (destinație) se obține pritr-o operație de adunare. Se adună la registrul sursă (destinație) indexul sursă (destinație) codificat binar pe 16 biți şi plasat în memorie imediat după codul instrucțiunii. Dacă instrucțiunea are adresare indexată atât la sursă cât şi la destinație (ca în figura de mai sus), atunci indexul sursă va fi plasat imediat după codul instrucțiunii (la adresa PC+2) iar indexul destinație va fi plasat la PC+4. La PC+6 se va găsi codul următoarei instrucțiuni. Dacă instrucțiunea are adresare indexată doar la unul dintre operanzi (fie sursă, fie destinație), atunci indexul (sursă sau destinație) va fi plasat imediat după codul instrucțiunii (la adresa PC+2). La adresa PC+4 va fi codul următoarei instrucțiuni.

Observație: În cazul instrucțiunilor cu doi operanzi modurile de adresare pot apărea în orice combinație (ortogonalitate). Exemple:

- mod adresare imediat la operandul sursă, mod adresare registru direct la operandul destinație
- mod adresare registru direct la operandul sursă, mod adresare indexat la operandul destinație
- mod adresare registru indirect la operandul sursă, mod adresare imediat la operandul destinație

D. Conținutul proiectului

Proiectul va conține următoarele capitole:

1. Introducere (prezentarea temei)
2. Definirea formatului microinstrucțiunii
3. Proiectarea microsecvențiatorului (automatul SEQ)
4. Proiectarea blocurilor BGC-ului microprogramat:
    - decodificatoarele de microcomenzi
    - blocul de selecție index
    - blocul de generare a funcției globale de ramificație (g).
5. Proiectarea microprogramului de emulare a setului de instrucțiuni