DOCUMENTATIE

PROCESOR MIPS

IRIMES DAVID

GR. 7

LISTA INSTRUCTIUNI (tip R | tip I | tip J):

Sintaxa ASM	Descriere RTL	Format
add $d, $s, $t	$d <- $s + $t; PC <- PC + 4;	000000 sssss ttttt ddddd 00000 100000
sub $d, $s, $t	$d <- $s - $t; PC <- PC + 4;	000000 sssss ttttt ddddd 00000 100010
sll $d, $t, h	$d <-$t << h; PC <-PC + 4;	000000 00000 ttttt ddddd hhhhh 000000
srl $d, $t, h	$d <-$t >> h; PC <-PC + 4;	000000 00000 ttttt ddddd hhhhh 000010
and $d, $s, $t	$d <-$s & $t; PC <-PC + 4;	000000 sssss ttttt ddddd 00000 100100
or $d, $s, $t	$d <-$s | $t; PC <-PC + 4;	000000 sssss ttttt ddddd 00000 100101
slt $d, $s, $t	PC <-PC + 4; if $s < $t then $d <-1 else $d <-0;	000000 sssss ttttt ddddd 00000 101010
xor $d, $s, $t	$d <-$s ^ $t; PC <-PC + 4;	000000 sssss ttttt ddddd 00000 100110
addi $t, $s, imm	$t <-$s + SE(imm); PC <-PC +4;	001000 sssss ttttt iiiiiiiiiiiiiiii
andi $t, $s, imm	$t <-$s & ZE(imm); PC <-PC +4;	001100 sssss ttttt iiiiiiiiiiiiiiii
lw $t, offset($s)	$t <-MEM[$s + SE(offset)]; PC <-PC + 4;	100011 sssss ttttt oooooooooooooooo
sw $t, offset($s)	MEM[$s + SE(offset)] <- $t; PC <-PC + 4;	101011 sssss ttttt oooooooooooooooo
beq $s, $t, offset	if $s = $t then PC <- (PC + 4) + (SE(offset) << 2) else PC <-PC + 4;	000100 sssss ttttt oooooooooooooooo
bne $s, $t, offset	if $s ≠ $t then PC <- (PC + 4) + (SE(offset) << 2) else PC <-PC + 4;	000101 sssss ttttt oooooooooooooooo
slti $t, $s, imm	PC <- PC + 4; if $s < SE(imm) then $t <- 1 else $t <- 0;	001010 sssss ttttt iiiiiiiiiiiiiiii
j addr	if $s ≠ $t then PC <- (PC + 4) + (SE(offset) << 2) else PC ⇓ PC + 4;	000010 aaaaaaaaaaaaaaaaaaaaaaaaaa

COD ASM-MIPS:

```
.data
arr: .word 5,1,3,2,12,10,5,6,20,22,20,6,10,20
size: .word 14

.text
.globl main

main:
 # Inițializăm dimensiunea si indexul
    lw $a0, size(&zero) #size
    lw $a1, 0($zero) #indice
    lw $a3, 0($zero) # max = $a3 = 0 initial
    lw $a4, 0($zero) # RES = cnt_max = $a4 = 0 -initial

calcResult:
    continue:
    beq $a1, $a0, end #terminarea buclei
    lw $a2, arr($a1) # element curent
    addi $a1,$a1,1
    j isMultipleof5

    process:
    beq $v0, $zero, continue

    #cod pentru contorizarea numarului maxim divizibil cu 5
    slt $s1, $a3, $a2
    bne $s1, $zero, new_max
    beq $a2, $a3, increment_cnt
      j continue
    increment_cnt:
      addi $a4,$a4,1
      j continue

    new_max:
    andi $a4, $a4, 1 #resetam la 1 contorul
    add $a3, $zero, $a2
    j continue

isMultipleof5:
    beq $a2,$zero, isMultipleof5_true
    add $s0, $zero, $a2 # Salvăm elem. curent în $s0
    loop:
    addi $s0, $s0, -5
    beq $s0,$zero, isMultipleof5_true
    slti $s1,$s0,5
    bne $s1,$zero,isMultipleof5_false
    j loop

isMultipleof5_true:
    # Setăm rezultatul în $v0
    addi $v0, $zero, 1
    j process
```

```
isMultipleof5_false:
    # Setăm rezultatul în $v0
    addi $v0, $zero, 0
    j process

end:
# Terminare program
```

COD MASINA (reprezentare si in HEXA in partea dreapta):

B"100011_00000_10000_0000000000000001", --8C100001 --lw $a0, size(&zero) #size

B"100011_00000_10001_0000000000000000", --8C110000 --lw $a1, 0($zero) #indice

B"100011_00000_10011_0000000000000000", --8C130000 --lw $a3, 0($zero) # max = $a3 = 0 initial

B"100011_00000_10100_0000000000000000", --8C140000 --lw $a4, 0($zero) # RES = cnt_max = $a4

B"000100_10001_10000_0000000000011000", --12300018 --beq $a1, $a0, end #terminarea buclei

B"100011_10001_10010_0000000000000010", --8E320002 --lw $a2, arr($a1) # element curent

B"001000_10001_10001_0000000000000001", --22310001 --addi $a1, $a1,1

B"000010_00000000000000000000010010", --20000012 --j isMultipleof5

B"000100_00001_00000_1111111111111011", --1020FFFB --beq $v0, $zero, continue

B"000000_10011_10010_11001_00000_101010", --0272C82A --slt $s1, $a3, $a2

B"000101_11001_00000_0000000000000100", --17200004 --bne $s1, $zero, new_max

B"000100_10010_10011_0000000000000001", --12530001 --beq $a2, $a3, increment_cnt

B"000010_00000000000000000000000100", --20000004 --j continue

B"001000_10100_10100_0000000000000001", --22940001 --addi $a4, $a4,1

B"000010_00000000000000000000000100", --20000004 --j continue

B"001100_10100_10100_0000000000000001", --32940001 --andi $a4, $a4, 1 #resetam la 1 contorul

B"000000_00000_10010_10011_00000_100000", --00129820 --add $a3, $zero, $a2

B"000010_00000000000000000000000100", --20000004 --j continue

B"000100_10010_00000_0000000000000110", --12400006 --beq $a2, $zero, isMultipleof5_true

B"000000_00000_10010_11000_00000_100000", --0012C020 -- add $s0, $zero, $a2

B"000000_11000_11000_1111111111111011", --318FFFB --addi $s0, $s0, -5

B"000100_11000_00000_0000000000000100", --13000004 --beq $s0, $zero, isMultipleof5_true

B"001010_11000_11001_0000000000000101", --2B190005 --slti $s1, $s0,5

B"000101_11001_00000_0000000000000011", --17200003 --bne $s1, $zero, isMultipleof5_false

B"000010_00000000000000000000010100", --20000014 -- j loop

B"001000_00000_00001_0000000000000001", --20010001 --addi $v0, $zero, 1

B"000010_00000000000000000000001000", --20000008 --j process

B"001000_00000_00001_0000000000000000", --20010000 --addi $v0, $zero, 0

B"000010_00000000000000000000001000", --20000008 --j process

LEGENDA:

size = 0000000000000001 | arr = 0000000000000010

$zero= 0000000000000000

$a0= 10000 | $a1= 10001 | $a2= 10010 | $a3= 10011 | $a4= 10100 --rezultat stocat aici

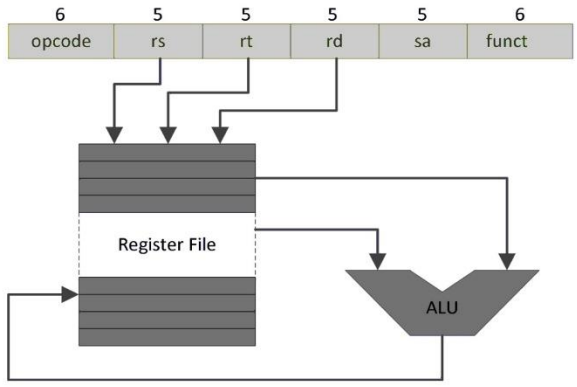
$v0= 00001 | $s0= 11000 | $s1= 11001

TABEL SEMNALE DE CONTROL:

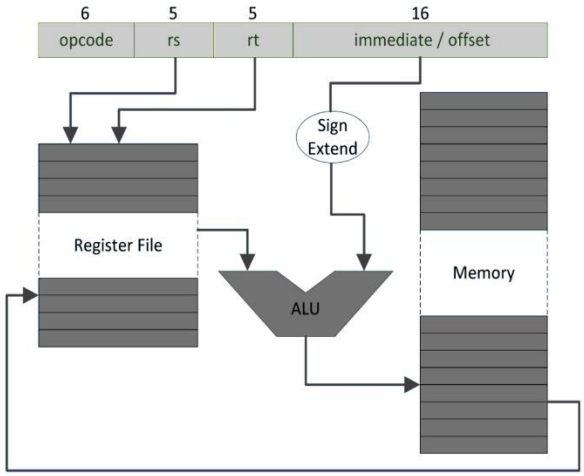
Instructiune	Reg Dst	Reg Write	ALU Src	Ext Op	ALU Ctrl	ALU Op	Mem Write	Memto Reg	Bne	Branch	Jump
add (tip R)	1	1	0	X	000 (+)	000	0	0	0	0	0
sub (tip R)	1	1	0	X	001 (-)	000	0	0	0	0	0
or (tip R)	1	1	0	X	101(|)	000	0	0	0	0	0
and (tip R)	1	1	0	X	100 (&)	000	0	0	0	0	0
xor	1	1	0	X	110(^)	000	0	0	0	0	0
sll	1	1	0	X	010(<<)	000	0	0	0	0	0
srl	1	1	0	X	011(>>)	000	0	0	0	0	0
slt	1	1	0	X	111(comp)	000	0	0	0	0	0
addi	0	1	1	1	000 (+)	001	0	0	0	0	0
andi	0	1	1	0	100 (&)	010	0	0	0	0	0
lw	0	1	1	1	000 (+)	011	0	1	0	0	0
sw	X	0	1	1	000 (+)	100	1	X	0	0	0
beq	X	0	0	1	001 (-)	101	0	X	0	1	0
bne	X	0	0	1	001(-)	110	0	X	1	0	0
slti	X	0	0	0	111(comp)	111	0	X	0	0	0
j	X	0	X	X	XXX	XXX	0	X	X	X	1

DIAGRAME PROCESARE (add, lw, beq, j)

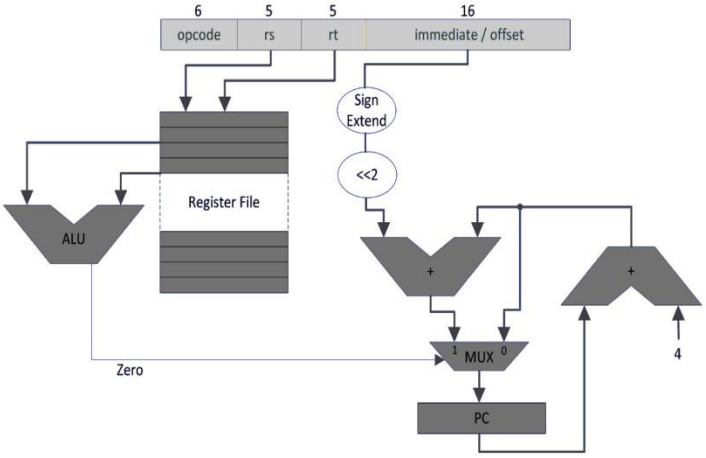
add



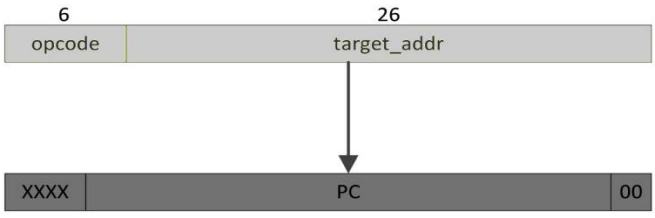
lw



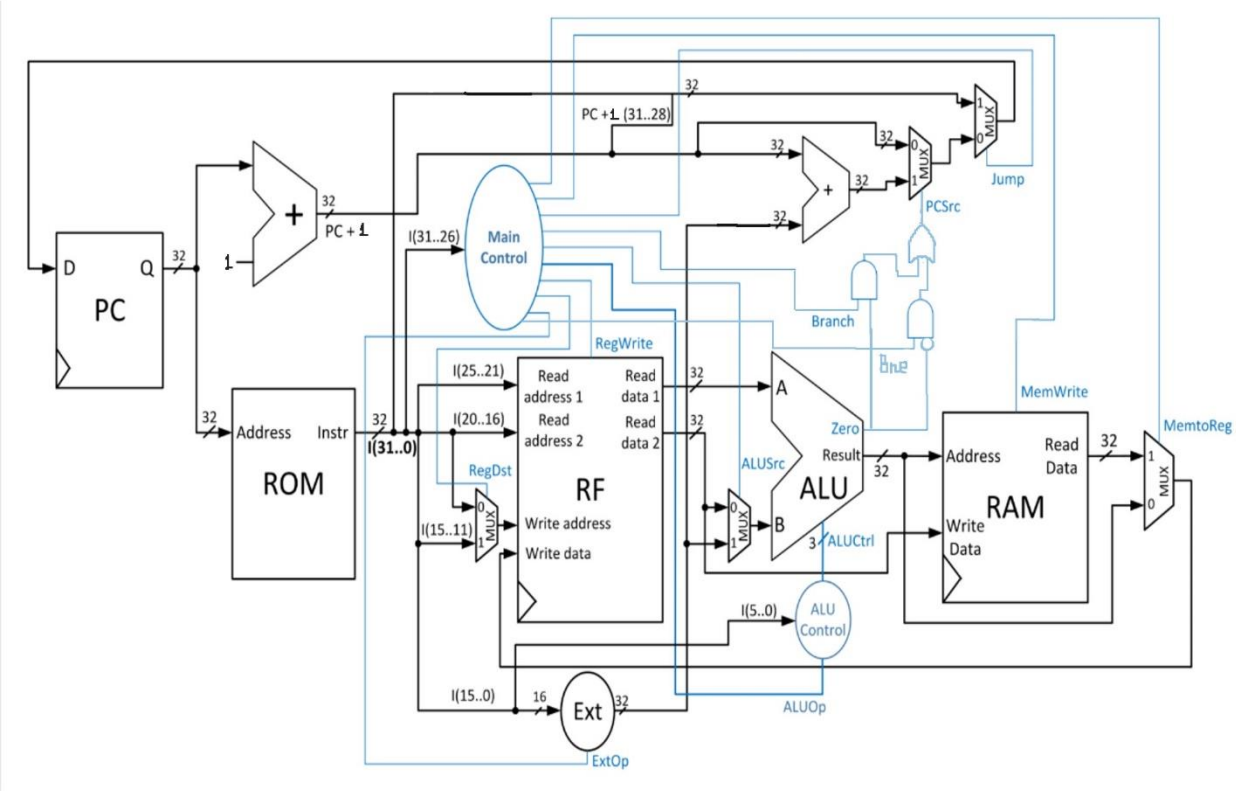
bec


6
5
5
16
opcode
rs
rt
immediate / offset
Sign Extend
<<2
Register File
ALU
+
+
1
MUX
0
4
Zero
PC

j


6
26
opcode
target_addr
XXXX
PC
00

SCHEMA COMPLETA:



DESCRIERE FUNCTIONALITATE:

- Avem un array de numere si trebuie sa contorizam aparitia celui mai mare multiplu de 5. Rezultatul se va retine in registrul $a4, iar determinarea daca este un numar divizibil cu 5 sau nu se va face prin scadere repetata.
- Se initializeaza variabilele si se pregateste pentru procesare array-ul.
- In "calcResult", se itereaza prin array si se determina cel mai mare numar divizibil cu 5, contorizandu-l.

— Funcția "isMultipleof5" verifica daca un element este divizibil cu 5. Se intra într-o bucla in care se scade 5 din valoarea elementului curent pana cand aceasta devine 0 sau mai mic decat 5
— Cand am ajuns sa procesam si ultimul element, contorul de bucla va fi egal cu marimea array-ului si se va termina executia.

DESCRIERE INSTRUCTIUNI NOI:

1. slt (Set on Less Than): Această instrucțiune compară două valori și setează un registru la 1 dacă prima valoare este mai mică decât a doua și la 0 în caz contrar.
2. xor (exclusive OR): Această instrucțiune realizează operația logică XOR între două registre. Rezultatul este stocat într-un al treilea registru.
3. slti (Set on Less Than Immediate): Similar cu slt, această instrucțiune compară un registru cu o valoare imediată și setează un alt registru la 1 dacă registrul este mai mic decât valoarea imediată și la 0 în caz contrar.
4. andi (AND Immediate): Această instrucțiune realizează operația logică AND între un registru și o valoare imediată și stochează rezultatul într-un alt registru.
5. bne (Branch on Not Equal): Această instrucțiune compară două registre și sare la o anumită adresă dacă acestea nu sunt egale. Dacă sunt egale, execuția continuă cu următoarea instrucțiune din fluxul normal.

SCHEMATIC RTL: