Suma numerelor pare de pe poziții impare

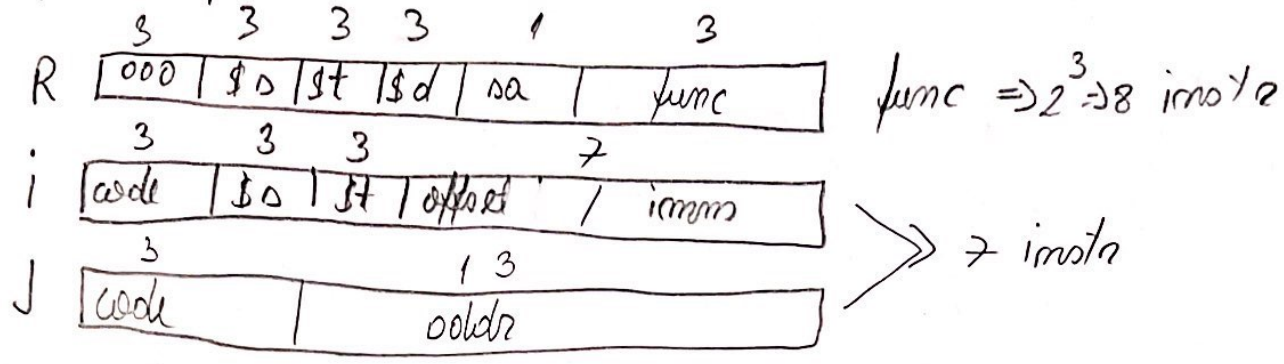
Cod C:

```
int A[10] = {2, 3, 4, 7, 9, 3, 1, 8, 6, 9 };
int sum=0;
for(int i=0; i<10; i++)
 { if (i%2 != 0)
      if (A[i] %2 == 0)
          sum = sum + A[i];
 }
```

- final A se află în memorie, primul element la adresa A_addr
- sum stocher momentan în reg $5
- contorul în reg $1
- pt a accesa fiecare elem. se va folosi un index $2 ce va crește din 1 în 1
- pt a compara contorul cu 10 se va lua $4

```
0.  addi $6, $0, 1
1.  add  $1, $0, $0  // contorul
2.  addi $4, $0, 10  // nr max de iterații
3.  add  $2, $0, $0  // index loc de memorie
4.  add  $5, $0, $0  // sum = 0
5.  begin_loop: beq $1, $4, end_loop
6.  lw   $3, A_addr($2), 0
7.  and $7, $1, $6  // și pe biți între index și 1
8.  bne $7, $6, loop2  // dacă e par, sare, $7 tot nu își 1
9.  and $7, $3, $6  // și pe biți între element și 1
10. bne $7, $0, loop2
11. add $5, $5, $3,  // sum = sum + a[i]
12. loop2: addi $2, $2, 1  // index element următor
13. addi $1, $1, 1  // i++
14. j begin_loop
15. end_loop: nop $5,
      sum_addr($a), 0
```

Mips 16 - format instructiuni



Instructiuni : func:

R: add 011
sub 001
sll 100
srl 101
srlv 000
or 010
xor 110
xnor 111

i: beq 110
bne 101
sw 100
addi 001
lw 010

J: j 111

Cod mașină:

0. 001_000_110_0000001 — 2301
1. 000_000_000_001_0_011 — 0013
2. 001_000_100_0001010 — 220A
3. 000_000_000_010_0_011 — 0023
4. 000_000_000_101_0_011 — 0053
5. 110_001_100_0001001 — D089
6. 101_010_011_0000000 — A980
7. 000_001_110_111_0_000 — 0770
8. 010_111_110_0000011 — 5F03
9. 000_011_110_111_0_000 — 0F70
10. 010_111_000_0000001 — DC01
11. 000_101_001_101_0_011 — 15D3
12. 001_010_010_0000001 — 2901
13. 001_001_001_0000001 — 2481
14. 111_0000000000101 — E005
15. 100_000_101_0000000 — 8280

SEMNALE DE CONTROL

Instr	Opcode	RegDst	ExtOp	AluSrc	Branch	Jump	AluOp	MemWr	MemtoReg	RegWrite	Func	ALUCTRL
add	000	1	0	0	0	0	000	0	0	1	000	000
sub	000	1	0	0	0	0	000	0	0	1	001	001
and	000	1	0	0	0	0	000	0	0	1	011	011
or	000	1	0	0	0	0	000	0	0	1	010	010
sll	000	1	0	0	0	0	000	0	0	1	100	100
srl	000	1	0	0	0	0	000	0	0	1	101	101
xor	000	1	0	0	0	0	000	0	0	1	110	110
xnor	000	1	0	0	0	0	000	0	0	1	111	111
addi	001	0	1	1	0	0	001	0	0	1	-	000
lw	101	0	1	1	0	0	001	0	1	1	-	000
sw	100	0	1	1	0	0	001	1	0	0	-	000
beq	110	0	1	0	1	0	010	0	0	0	-	001
bne	010	0	1	0	1	0	010	0	0	0	-	001
bgtz	011	0	1	0	1	0	010	0	0	0	-	001
j	111	0	0	0	0	1	-	0	0	0	-	-

TRASAREA EXECUȚIEI

PAS	SW(7:5) / Instrucțiune	000 / Imediat	001 / EIMC	010 / RD1	011 / RD2	100 / ALURES	101 / ExtImm	110 / Ext_func	111 / WA	BranchAdd	JumpAdd	
0	oddi	2301	0001	0000	0101	0001	0001	0000	0001	–	–	
1	add	0013	0002	0000	0000	0000	0013	0000	0000	–	–	
2	oddi	220A	0003	0000	0000	000A	000A	0000	000A	–	–	Primo iterație
3	add	0023	0004	0000	0000	0000	0023	0000	0000	—	—	
4	add	0053	0005	0000	0000	0000	0053	0000	0000	-	-	
5	beq	D089	0006	000A	0000	000A	0009	0000	000A	8280	–	
6	lw	A980	0007	0000	0000	0000	0000	0000	0002	-	–	
7	and	0770	0008	0000	0001	0000	0070	0000	0000	-	-	
8	bne	5F03	0009	0000	0001	FFFF	0003	0000	FFFF	2901	—	→
9	and	0F70	000A	0004	0001	0000	0070	0000	0000	—	–	
10	bne	5C01	000B	0000	0000	0000	0001	0000	0000	2901	-	A doua iterație
11	add	1513	000C	0000	0004	0004	0053	0000	0004	–	–	
12	addi	2901	000D	0001	0001	0002	0001	0000	0002	–	–	
13	addi	2481	000E	0001	0001	0002	0001	0000	0002	–	–	
14	j	E005	0007	0000	0000	0000	0005	0000	0000	–	8089	
15	sw	8280	0010	0000	0010	0000	0000	0000	0000	–	–	→ ultima

SCHEMA MIPS 16