Documentatie

Mips ciclu unic 32 de biti

Nume: Saca Victor-Valentin

Setul de instructiuni utilizat, cu codul RTL corespunzator

sw

add $d, $s, $t : d = s + t ; 000000 sssss ttttt ddddd 00000 000000

sub $d, $s, $t : d = s - t, pc += 4 000000 sssss ttttt ddddd 00000 100001

sll $d, $t, h : $d = $t << h, pc+= 4 ; 000000 00000 ttttt ddddd hhhhh 000010

srl $d, $t, h : d = t << h, pc += 4 ; 000000 00000 ddddd ttttt hhhhh 000011

and $d, $s, $t : d = s and t ; 000000 sssss ttttt ddddd 00000 000100

or $d, $s, $t : d = s and t, PC += 4 ; 000000 sssss ttttt ddddd 00000 000101

slt $d, $s, $t : if s < t then d = 1 else d = 0 ; 000000 sssss ttttt ddddd 00000 000110

xor $d, $s, $t : $d = $s + $t; pc += 4; : 000000 sssss ttttt ddddd 00000 100110

I

addi $t, $s, imm : t = s + SE(imm) ; 001000 sssss ttttt iiiiiiiiiiiiiiii

lw $t, offset($s) : t = mem[s + SE(offset)]; pc += 4 ; 100011 sssss ttttt oooooooooooooooo

sw $t, offset($s) ; 101011 sssss ttttt oooooooooooooooo

beq $s, $t, offset : if s == t then pc += 4 + (SE(offset) << 2) else pc +=4 ; 000100 sssss ttttt oooooooooooooooo

andi $t, $s, imm : $t = $s & SE(imm); pc += 4 : 001100 sssss ttttt iiiiiiiiiiiiiiii

ori $t, $s, imm : $t = $s | ZE(imm); PC+=4 : 001101 sssss ttttt iiiiiiiiiiiiiiii

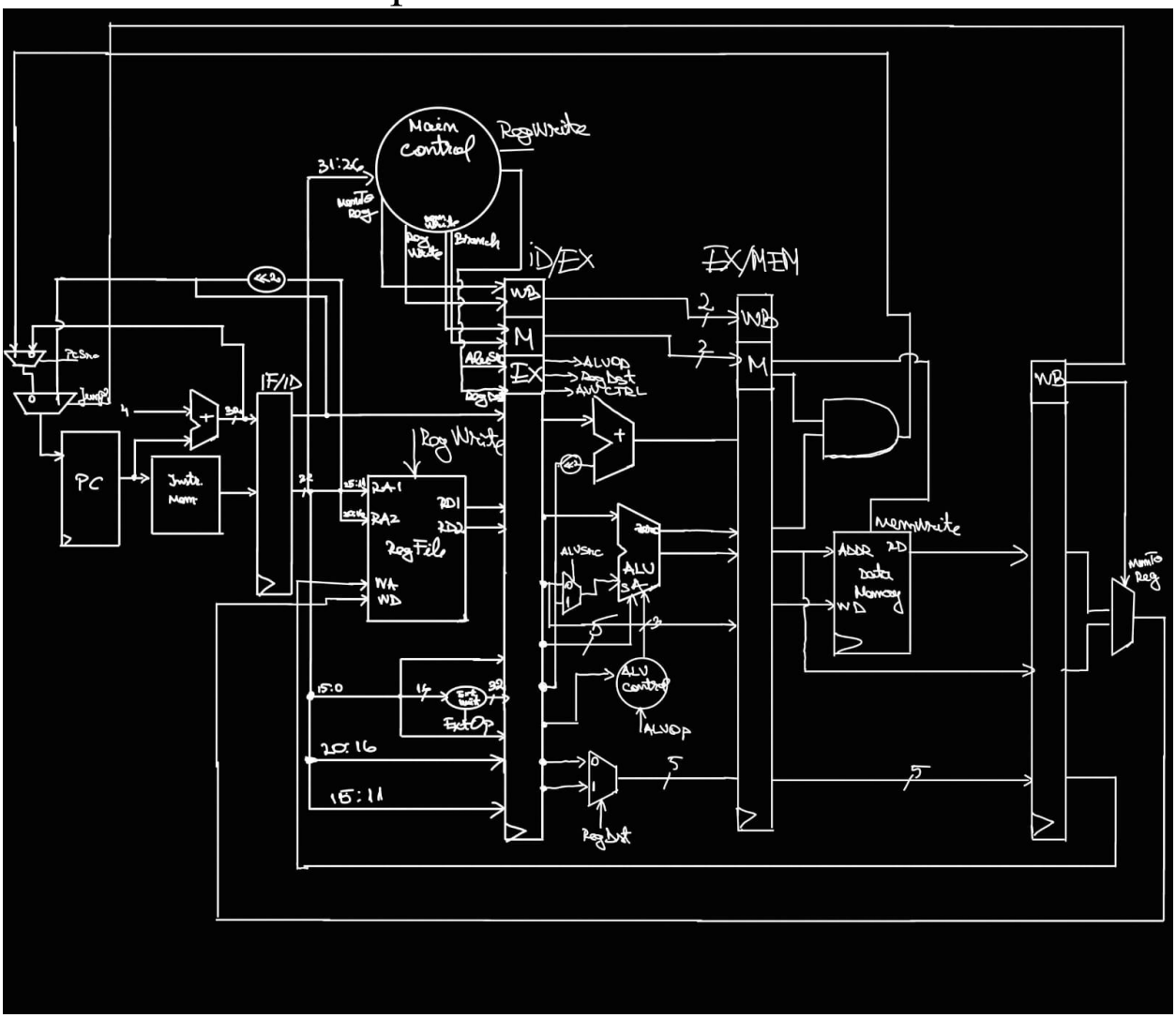
J

j addr : pc = (pc + 4)[31:28] || (addr << 2) ; 000010 aaaaaaaaaaaaaaaaaaaaaaaaaa

Setul de semnale de control:

instr	regDst	extOp	aluSrc	Branch	Jump	aluOp	memWrite	memReg	regWrite	branchnz
add	0	1	0	0	0	000	0	0	1	0
Addi	0	1	1	0	0	001	0	0	1	0
Lw	0	1	1	0	0	000	0	1	1	0
Sw	X	1	1	0	0	000	1	X	0	0
Beq	X	1	0	1	0	001	0	X	0	0
Ori	X	0	1	0	0	011	0	X	1	0
Andi	0	0	1	0	0	100	0	0	1	0

Schema detaliata a proiectului:



Programul de testare implementat: Generare sirul lui fibonacci

In cod assembly mips32:

```c
int i = 0;
int* ram = (int*) malloc(1000*sizeof(int));
*ram = 0;
ram++;
*(ram) = 1;
ram++;
*(ram) = 1;
ram++;

while(*(ram - 1) > INT_MIN / 2){
    *ram = *(ram - 1) + *(ram - 2);
    printf("%d\n", *ram);
    ram++;
    i++;
}


return 0;
```

```
ADDI $1, $0, 0
ADDI $2, $0, 1
ADDI $3, $0, 0
ADDI $4, $0, 4
NOOP
SW $1, 0($3)
SW $2, 0($4)
LW $1, 0($3)
LW $2, 0($4)
NOOP
NOOP
ADD $5, $1, $2
ADD $1, $0, $2
NOOP
ADD $2, $0, $5
J 11
NOOP
```

Programul in cod masina:

```
B"001000_00000_00001_0000000000000000",     -- X"20010000",
B"001000_00000_00010_0000000000000001",     -- X"20020001",
B"001000_00000_00011_0000000000000000",     -- X"20030000",
B"001000_00000_00100_0000000000000100",     -- X"20040004",
B"000000_00000_00000_0000000000000000",     -- X"00000000",
B"101011_00011_00001_0000000000000000",     -- X"AC610000",
B"101011_00100_00010_0000000000000000",     -- X"AC820000",
B"100011_00011_00001_0000000000000000",     -- X"8C610000",
B"100011_00100_00010_0000000000000000",     -- X"8C820000",
B"000000_00000_00000_0000000000000000",     -- X"00000000",
B"000000_00000_00000_0000000000000000",     -- X"00000000",
B"000000_00001_00010_00101_00000_100000",   -- X"00222820",
B"000000_00000_00010_00001_00000_100000",   -- X"00020820",
B"000000_00000_00000_00000_00000_000000",   -- X"00000000",
B"000000_00000_00101_00010_00000_100000",   -- X"00051020",
B"000010_00000000000000000000001011",       -- X"0800000B",
B"000000_00000000000000000000000000",
```

Hazard de date:

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
ADDI $1, $0, 0	IF	ID($0)	EX	MEM	WB($1)															
ADDI $2, $0, 1		IF	ID($0)	EX	MEM	WB($2)														
ADDI $3, $0, 0			IF	ID($0)	EX	MEM	WB($3)													
ADDI $4, $0, 4				IF	ID($0)	EX	MEM	WB($4)												
NOOP					IF	ID	EX	MEM	WB											
SW $1, 0($3)						IF	ID($3)	EX	MEM	WB($1)										
SW $2, 0($4)							IF	ID($4)	EX	MEM	WB($2)									
LW $1, 0($3)								IF	ID($3)	EX	MEM	WB($1)								
LW $2, 0($4)									IF	ID($4)	EX	MEM	WB($2)							
NOOP										IF	ID	EX	MEM	WB						
NOOP											IF	ID	EX	MEM	WB					
ADD $5, $1, $2												IF	ID($1,2)	EX	MEM	WB($5)				
ADD $1, $0, $2													IF	ID($0,2)	EX	MEM	WB($1)			
NOOP														IF	ID	EX	MEM	WB		
ADD $2, $0, $5															IF	ID($0, 5)	EX	MEM	WB($2)	
J 11																IF	ID	EX	MEM	WB
NOOP																				

Registrii

Configurare registre MIPS32 Pipeline – varianta 1

Se introduc pe coloane semnalele de date și control mapate la registre, de sus în jos, începând de la biții cei mai semnificativi ai registrului către cei mai puțin semnificativi. Se introduc în paranteză biții din registru alocați pentru fiecare semnal în parte. În dreptul numelui registrelor din primul rând se înlocuiește în paranteză semnul ? cu poziția bitului cel mai semnificativ.

REG_IF_ID[63:0]	REG_ID_EX[?:0]	REG_EX_MEM[?:0]	REG_MEM_WB[?:0]
Instruction [63:32]	RegDst [<poziția_în_reg>]	Branch [<poziția_în_reg>]	RegWr [<poziția_în_reg>]
PCp4 [31:0]	Instr(15:11) : [157 : 153]	InstrMux(20:11) : [105 : 101]	InstrMux : [70 : 66]
	Instr(20:16) : [152 : 148]	RD2 : [100 : 69]	aluRes : [65 : 34]
	Instr(5:0) : [147 : 142]	aluRes : [68 : 37]	RD : [33 : 2]
	Instr_ext: [141 : 110]	Zero : [36]	WB : [1 : 0]
	Instr(10:6) : [109 : 105]	branchInst : [35 : 4]	
	RD2 : [104 : 73]	M : [3 : 2]	
	RD1 : [72 : 41]	WB : [1:0]	
	PCp4 : [40 : 9]		
	EX(aluOp, aluSrc, regDST) : [8:4]		
	M(memwrite, branch) : [3:2]		
	WB(MemToReg, RegWrite)[1:0]		