PROIECT MIPS CICLU UNIC

Nemes Teodora

Grupa 30226

1. Tabel semnale de control

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Intructiune | PC\_Src | Jump | Branch | Reg\_Dst | Reg\_Write | Ext\_op | Mem\_Write | Mem\_to\_reg | ALU\_Src |
| Add | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Sub | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Sll | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Srl | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| And | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Or | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Xor | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Sllv | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Addi | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| Lw | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| Sw | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| Beq | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| Ori | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| Xori | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| J | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Descrierea instructiunilor
   1. addi $1 $2 3: adaugă valoarea imediată 3 la conținutul registrului $2 și stochează rezultatul în registru $1.
   2. addi $3 $4 2: adaugă valoarea imediată 2 la conținutul registrului $4 și stochează rezultatul în registrul $3
   3. add $2 $1 $3: realizează adunarea dintre conținutul registrului $1 și conținutul registrului $3 și stochează rezultatul în registrul $2
   4. sub $4 $2 $3: realizează scăderea dintre conținutul registrului $3 din conținutul registrului $2 și stochează rezultatul în registrul $4
   5. sll $5 $2 1: realizează o operație de shiftare la stânga logică a conținutului registrului $2 cu o valoare imediată de 1 bit și stochează rezultatul în registrul $5.
   6. srl $4 $1 1: realizează o operație de shiftare la dreapta logică a conținutului registrului $1 cu o valoare imediată de 1 bit și stochează rezultatul în registrul $4
   7. and $1 $2 $3: realizează o operație logică "AND" între conținuturile registrilor $2 și $3 și stochează rezultatul în registrul $1
   8. or $2 $1 $4: realizează o operație logică "OR" între conținuturile registrilor $1 și $4 și stochează rezultatul în registrul $2
   9. xor $3 $2 $4: realizează o operație logică "XOR" între conținuturile registrilor $2 și $4 și stochează rezultatul în registrul $3.
   10. sllv $3 $2 $1: realizează o operație de shiftare la stânga logică variabilă unde numărul de biți cu care se realizează shiftarea este dat de conținutul registrului $1, iar valorile din registrul $2 sunt shiftate la stânga. Rezultatul este stocat în registrul $3.
   11. lw $1 3($2): realizează încărcarea unui cuvânt (word) din memorie în registrul $1, cu o compensație de adresă de 3 de la conținutul registrului $2
   12. sw $5 2($3): realizează stocarea conținutului registrului $5 în memorie, la o adresă deplasată cu 2 octeți față de conținutul registrului $3
   13. beq $2 $3 2: compară conținutul registrului $2 cu conținutul registrului $3, iar dacă acestea sunt egale, se realizează un salt relativ la o adresă situată la 2 instrucțiuni distanță de instrucțiunea curentă
   14. j 1: realizează un salt absolut la o adresă specificată, în acest caz, adresa 1
   15. ori $1 $2 4: realizează o operație logică "OR" între conținutul registrului $2 și valoarea imediată 4 și stochează rezultatul în registrul $1
   16. xori $5 $3 2: realizează o operație logică "XOR" între conținutul registrului $3 și valoarea imediată 2 și stochează rezultatul în registrul $5
2. Infomatii relevante

* Proiectul se regaseste intr-un stadiu functional, este implementat in totalitate
* Momentan nu am reusit sa generez bitstream-ul datorita unei erori interne din versiunea Vivado instalata pe laptop-ul personal

1. Instructiuni suplimentare
   1. Xor

* Instructiune: xor $3 $2 $4
* Reprezentare in binar: 000000\_00010\_00100\_00011\_00000\_100111
* RTL abstract: RF[$3] <- RF[$2] ^ RF[$4]
  1. Descriere functionalitate: realizează o operație logică "XOR" între conținuturile registrilor $2 și $4 și stochează rezultatul în registrul $3.
  2. Sllv
* Instructiune: sllv $3 $2 $1
* Reprezentare in binar: 000000\_00010\_00001\_00011\_00010\_101000
* RTL abstract: RF[$3] <- RF[$2] << RF[$1]
* Descriere functionalitate: realizează o operație de shiftare la stânga logică variabilă unde numărul de biți cu care se realizează shiftarea este dat de conținutul registrului $1, iar valorile din registrul $2 sunt shiftate la stânga. Rezultatul este stocat în registrul $3.
  1. Ori
* Instructiune: ori $1 $2 4
* Reprezentare in binar: 001011\_00010\_00001\_0000000000000100
* RTL abstract: RF[$1] <- RF[$2] | Z\_imm(4)
* Descriere functionalitate: realizează o operație logică "OR" între conținutul registrului $2 și valoarea imediată 4 și stochează rezultatul în registrul $1
  1. Xori
* Instructiune: xori $5 $3 2
* Reprezentare in binar: 001001\_00011\_00101\_0000000000000010
* RTL abstract: RF[$5] <- RF[$3] ^ Z\_ext(2)
  1. Descriere functionalitate: realizează o operație logică "XOR" între conținutul registrului $3 și valoarea imediată 2 și stochează rezultatul în registrul $5