Stephen Kairos Reyes

009862892

CS 47

Homework 2

|  |
| --- |
| 1. .include "cs47\_macro.asm" 2. .data 3. prompta: .asciiz "Enter +ve integer: " 4. promptb: .asciiz "Wrong input. Try again.\n" 5. resulta: .asciiz "Answer: " 6. resultb: .asciiz "Index " 7. resultc: .asciiz " fibonacci numbers are " 8. resultd: .asciiz ", " 9. newl: .asciiz "\n" 10. .text 11. begin: 12. print\_str(prompta) 13. read\_int($a1) 14. addi $t2, $zero, 2 15. div $a1, $t2 16. mflo $a1 17. mfhi $t2 18. print\_str(newl) 19. bltz $a1, bzzz 20. addi $t0, $zero, 0 21. addi $t1, $zero, 1 22. addi $sp, $sp, -4 23. sw $t0, ($sp) 24. addi $sp, $sp, -4 25. sw $t1, ($sp) 26. addi $a1, $a1, -1 27. fibloop: 28. add $t0, $t0, $t1 29. add $t1, $t1, $t0 30. addi $sp, $sp, -4 31. sw $t0, ($sp) 32. addi $sp, $sp, -4 33. sw $t1, ($sp) 34. addi $a1, $a1, -1 35. bgez $a1, fibloop 36. move $sp, $t1 37. done: 38. beqz $t2, other 39. print\_str(resulta) 40. print\_reg\_int($t1) 41. j news 42. other: 43. print\_str(resulta) 44. print\_reg\_int($t0) 45. news: 46. print\_str(newl) 47. print\_str(newl) 48. j begin 49. bzzz: 50. print\_str(promptb) 51. j begin |

|  |
| --- |
| 1. .include "cs47\_macro.asm" 2. .data 3. .align 2 4. var\_a: .word 2 3 5 5 8 10 11 17 18 20 5. var\_b: .word 5 6 7 8 14 15 17 6. var\_m: .word 10 7. var\_n: .word 7 8. var\_c: .word 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9. newl: .asciiz "\n" 10. .text 11. .globl start 12. start: 13. add $a1, $zero, $zero #index of var\_a 14. add $a2, $zero, $zero #index of var\_b 15. add $a3, $zero, $zero #index of var\_c 16. lw $s1, var\_m #length of var\_a 17. lw $s2, var\_n #length of var\_b 18. loopcheck: 19. sub $t0, $s1, $a1 #If index of a < length of a... 20. bgtz $t0, check #Jump to second check, else... 22. sub $t0, $s2, $a2 #If index of b < length of b... 23. bgtz $t0, addbtoc #Jump to add b to c as a is empty, else... 24. blez $t0, endProg #Since both arrays are empty, exit. 25. check: 26. sub $t0, $s2, $a2 #Since a is not empty, and if index of b < length of b... 27. bgtz $t0, looproutine #Jump to routine, else... 28. j addatoc #Since a is not empty, and b is empty, add a to c. 29. looproutine: 30. la $t1, var\_a #Load var\_a array into $t1. 31. mul $t0, $a1, 4 #Get pointer reference by mult index by 4. 32. add $t0, $t1, $t0 #Go to the index from pointer in array. 33. lw $t1, ($t0) #Get value from index in var\_a, store into $t1. 34. #Loaded val of var\_a at current index of $a1 into $t1 35. la $t2, var\_b #Load var\_b array into $t2. 36. mul $t0, $a2, 4 #Get pointer reference by mult index by 4. 37. add $t0, $t2, $t0 #Go to the index from pointer in array. 38. lw $t2, ($t0) #Get value from index in var\_b, store into $t2. 39. #Loaded val of var\_b at current index of $a2 into $t2 40. sub $t0, $t1, $t2 41. bgtz $t0, addbtoc 42. blez $t0, addatoc 43. #If a-b is greater than 0, a is bigger add b to c, else a is smaller, add a to c 45. addbtoc: 46. move $t4, $t2 47. add $a2, $a2, 1 #add 1 to index of var\_b 48. j addtoc 50. addatoc: 51. move $t4, $t1 52. add $a1, $a1, 1 #add 1 to index of var\_a 53. j addtoc 55. addtoc: 56. la $t3, var\_c 57. mul $t0, $a3, 4 58. add $t0, $t3, $t0 59. sw $t4, ($t0) #stores the value $t4 from the merge logic into $t3 ($t0) 60. add $a3, $a3, 1 #index of var\_c 61. j loopcheck 62. endProg: 63. li $v0, 10 64. syscall |

1. 2’s Complement, 610 and -310:
   1. 610 = 01102

-310 = 11012

* 1. 00000110

x 11111101

00000000110

00000000000

00000011000

00000110000

00001100000

00011000000

00110000000

+ 01100000000

101111011102  (convert to only 8-bits) -> 111011102 = -1810

* 1. Extend to 8-Bit:
     1. Zero Extend:

610 → 000001102 → 610

-310 → 000011012 → 1310

* + 1. Signed Extend:

610 → 000001102 → 610

-310 → 111111012 → -310

1. Truth Tables:
   * 1. F(x, y, z) = (xy)' + z

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **Z** | **XY** | **(XY)’** | **(XY)’+ Z** |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 |

* 1. F(x, y, z) = (x'yz') + (xy'z)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **X** | **Y** | **Z** | **X’** | **Y’** | **Z’** | **X’YZ’** | **XY’Z** | **F(X,Y,Z)** |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

1. Boolean Arithmatic Manipulation:
   1. x’y’z’ + xy’z’ + x’yz’ + xyz’ =

z’(x’y’ + xy’ + x’y + xy) =

z’((x + x’)y’ + (x + x’)y) =

z’((1)y’ + (1)y) =

z’(y’ + y) =

z’(1) = z’

* 1. (a’ + c)(a’ + d’)(b + c)(b + d’) =

(a’ + cd’)(b + cd’) =

a’b + cd’

1. Karnaugh Maps:
   1. f (A, B, C, D) = ∑ m (1, 2, 3, 4, 6, 7, 9, 11, 12, 13, 14, 15)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AB\CD** | **00** | **01** | **11** | **10** |
| **00** | 0 (0) | 1 (1) | 1 (3) | 1 (2) |
| **01** | 1 (4) | 0 (5) | 1 (7) | 1 (6) |
| **11** | 1 (C) | 1 (D) | 1 (E) | 1 (F) |
| **10** | 0 (8) | 1 (9) | 1 (B) | 0 (A) |

Prime Implicants: AD, BC, A’C, AB, CD, B’D, BD’

Essential Prime Implicants: BD’, B’D, A’C

SOP Expression: BD’ + B’D + A’C

* 1. f (w, x, y, z) = ∑ m (0, 5, 10, 15) + d(2, 7, 8, 13)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **WX\YZ** | **00** | **01** | **11** | **10** |
| **00** | 1 (0) | 0 (1) | 0 (3) | X (2) |
| **01** | 0 (4) | 1 (5) | X (7) | 0 (6) |
| **11** | 0 (C) | X (D) | 1 (E) | 0 (F) |
| **10** | X (8) | 0 (9) | 0 (B) | 1 (A) |

Prime Implicants: XZ, X’Z’

Essential Prime Implicants: XZ, X’Z’

SOP Expression: XZ + X’Z’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | Is Single Digit |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AB\CD** | **00** | **01** | **11** | **10** |
| **00** | 1 (0) | 1 (1) | 1 (3) | 1 (2) |
| **01** | 1 (4) | 1 (5) | 1 (7) | 1 (6) |
| **11** | 0 (C) | 0 (D) | 0 (E) | 0 (F) |
| **10** | 1 (8) | 1 (9) | 0 (B) | 0 (A) |

Essential Prime Implicants: B’C’, A’

Expression: A’ + B’C’

Converting Expression:

* A’ + B’C’ (Initial Expression)
* A’ + (B + C)’ (De Morgan’s Law)
* (A(B+C))’ (De Morgan’s Law)
* (AB + AC)’ (Distributive)
* (AB)’(AC)’ (De Morgan’s Law)
* Different Notation: (A NAND B) AND (A NAND C)
* AND can be rewritten as two NANDs

