MIPS project

מגיש : רום הירש

ת.ז : 313288763

חלק א' – המרת מחרוזת מספרית :

אופן פעולה של הפונקציה Ascii

1. טעינת מזיכרון של אות (כל אות בchar היא בית אחד)
2. בדיקת ערך האות האם הוא בין 0x40 ל- 0x30 שזה הערכים הascii שמייצגים digit (מספרים).
   a. במידה וזה בתחום נמיר את הספרה מascii לייצוג מספרי (ascii-0x30) ונשים אותה בערך העשרוני המתאים .
3. נחזור בלולאה על הפעולה עד לקבלת FALSE בתנאי (ערך האות בין 0x40 ל- 0x30) .
4. בסיום התוצאה נמצאת $0V

בדיקות :

1. בזיכרון : נזין בזיכרון ascii 412$7 – ניתן לראות שיצא כפי שנדרש 412=$v0

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x21081001	addi $8,$8,4097	5: ascii: addi $t0,$t0,0x1001 # $t0 = 0x1001
	0x00400004	0x00084400	sll $8,$8,16	6: sll $t0,$t0, 16 # $t0 = address of ascii string
	0x00400008	0x81090000	lb $9,0($8)	8: For: lb $t1,0($t0) # The current char address i'm checking (i + poi...
	0x0040000c	0x21080001	addi $8,$8,1	9: addi $t0,$t0,1 # load bite(char in index i (in asscii))
	0x00400010	0x292a0040	slti $10,$9,64	11: slti $t2,$t1,0x40 # if $t1 < 0x40
	0x00400014	0x200b0030	addi $11,$0,48	12: addi $t3,$0,0x30
	0x00400018	0x0169582a	slt $11,$11,$9	13: slt $t3,$t3,$t1 # if 0x30 < $t1
	0x0040001c	0x156a0005	bne $11,$10,5	14: bne $t3,$t2, END # if not 0x30 < $t1 <0x40 (ascii digit) jump t...
	0x00400020	0x2001000a	addi $1,$0,10	15: mul $v0,$v0,10 #
	0x00400024	0x70411002	mul $2,$2,$1	
	0x00400028	0x2129ffd0	addi $9,$9,-48	16: addi $t1,$t1,-0x30 # convert to digit number from ascii (ascii ...
	0x0040002c	0x00491020	add $2,$2,$9	17: add $v0,$v0,$t1 # $v0 += $t1
	0x00400030	0x08100002	j 0x00400008	18: j For

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	$ 2 1 4	\0 \0 \0 7	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010180	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☑ ASCII

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	10
$v0	2	412
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500996
$t1	9	36
$t2	10	1
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194356
hi		0
lo		410

2. בזיכרון : ascii: 23
$v0=23

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	10
$v0	2	23
$v1	3	0
$a0	4	0
$a1	5	0
$a2	6	0
$a3	7	0
$t0	8	268500995
$t1	9	0
$t2	10	1
$t3	11	0
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	0
$s2	18	0
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	0
pc		4194356
hi		0
lo		20

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x21081001	addi $8,$8,4097	5: ascii: addi $t0,$t0,0x1001 # $t0 = 0x1001
	0x00400004	0x00084400	sll $8,$8,16	6: sll $t0,$t0, 16 # $t0 = address of ascii string
	0x00400008	0x81090000	lb $9,0($8)	8: For: lb $t1,0($t0) # The current char address i'm checking (i + poi...
	0x0040000c	0x21080001	addi $8,$8,1	9: addi $t0,$t0,1 # load bite(char in index i (in asscii))
	0x00400010	0x292a0040	slti $10,$9,64	11: slti $t2,$t1,0x40 # if $t1 < 0x40
	0x00400014	0x200b0030	addi $11,$0,48	12: addi $t3,$0,0x30
	0x00400018	0x0169582a	slt $11,$11,$9	13: slt $t3,$t3,$t1 # if 0x30 < $t1
	0x0040001c	0x156a0005	bne $11,$10,5	14: bne $t3,$t2, END # if not 0x30 < $t1 <0x40 (ascii digit) jump t...
	0x00400020	0x2001000a	addi $1,$0,10	15: mul $v0,$v0,10 #
	0x00400024	0x70411002	mul $2,$2,$1	
	0x00400028	0x2129ffd0	addi $9,$9,-48	16: addi $t1,$t1,-0x30 # convert to digit number from ascii (ascii ...
	0x0040002c	0x00491020	add $2,$2,$9	17: add $v0,$v0,$t1 # $v0 += $t1
	0x00400030	0x08100002	j 0x00400008	18: j For

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	\0 \0 3 2	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010020	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010040	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100100e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010140	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010160	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x10010180	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0
0x100101e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0

0x10010000 (.data) ☑ Hexadecimal Addresses ☐ Hexadecimal Values ☑ ASCII

חלק ב' – סדרות Collatz

אופן פעולת התוכנית :

1. התוכנית מורכבת משני פונקציות main וcollatz.
2. תוכנית main:
   a. התוכנית רצה לולאה המעבירה כל פעם ערך רץ בין 1 ל – 100 ומכניסה אותו לפרמטר (num)$a0 שמקבלת הפונקציה collatz, ובכל ריצה קוראת לפונקציה collatz עם הערך num הנוכחי , וסכמת את הערך שהפונקציה collatz מחזירה (שזהו משתנה שסוכם את כמות סדרות collatz בעלות פחות מ 50 איברים) .
   b. בסיום הלולאה הפנקציה מדפיסה את סכום הסדרות שיש להם פחות מ- 50 איברים ומתחיל בספרה בין 1 – 100 , מדפיסה , ומס' ת.ז שלי.
3. תוכנית Collatz : מקבלת פרמטר ברגיסטר $a0 = num – המספר התחלתי של הסדרה
   a. התוכנית תעבור על כל איברי ספרת collatz בכל פעם תבדוק את התנאים הבאים (המעבר על כל האיברים נעשה בעזרת רקורסיה עד לתנאי סופי שnum שווה 1 או שcount ($s0) גדול מ 50 :
   b. if num == 1 תחזיר 1 ותסיים את התוכנית.
   c. אם num זוגי תקרא לפונקציה שנית אם שינוי בcollatz(num/2)
   d. אם num אי זוגי תקרא לפונקציה שנית אם שינוי בnum collatz(3*num + 1)
   e. בנוסף בכל כניסה לפונקציה המשנה הסופר count ($0S) יעלה באחד ומידה והוא גדול מ50 הפוקנציה תחזיר 0 .
   f. בכך בעצם התוכנית בודקת האם סדרת collatz עם משנה התחלתי שהוכנס num בעלת פחות איברים מ-50 (תחזיר אחד במידה וכן ואפס במידה ולא).

בדיקה : (לפי הרמז 4 קבוצות במספרים התחלתיים 1-10 :

```
4,313288763
-- program is finished running (dropped off bottom) --
```

תוצאה סופית :

Mars Messages | Run I/O

```
84,313288763
-- program is finished running (dropped off bottom) --
```

Clear