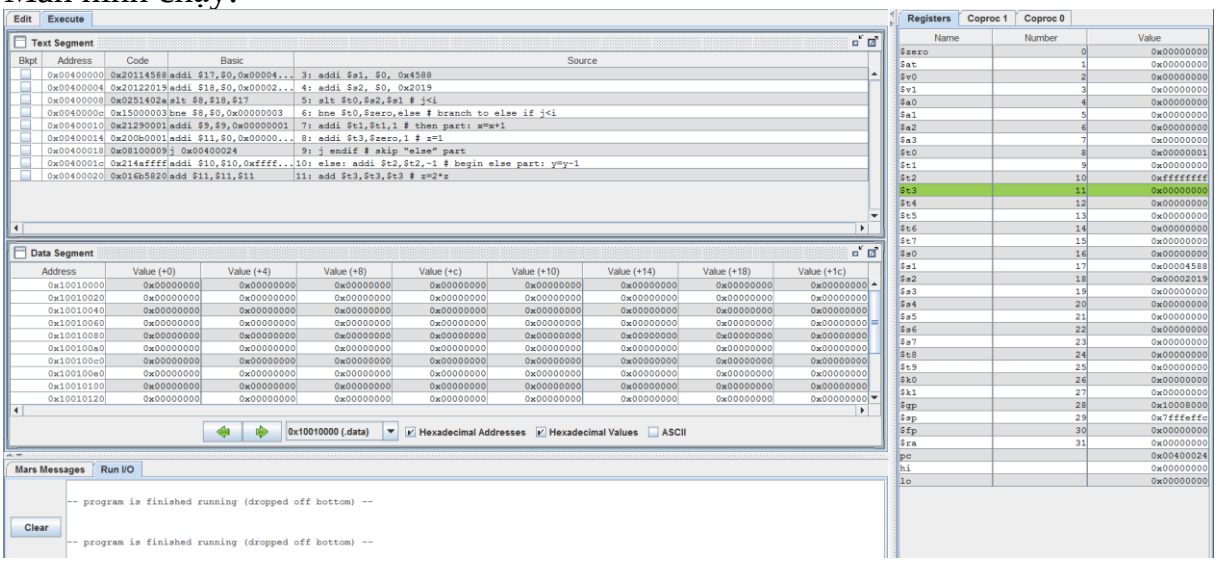
Báo cáo tuần 3

Assignment 1

- Khởi tạo: i =4588, j =2019
- Màn hình chạy:



- Ý nghĩa:
  if (not j<i) {x=x+1; z=1;} else {y=y-1; z=2*z;}

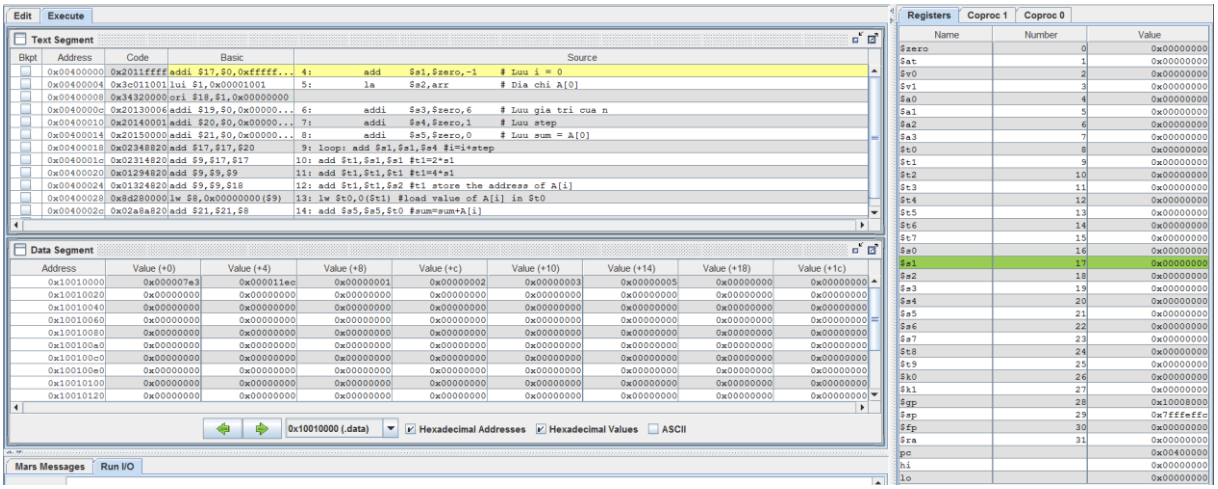
- Các bước chương trình:
  1. Step 3,4: Khởi tạo $s1 = 0x4588, $s2 =0x2019
  2. Step 5: slt so sánh nếu $s2 < $s1 thì $t0 = 1
  3. Step 6: bne tạo nhánh nếu $t0 khác 0 thì đến nhánh else
  4. Step 9:Do $t0 = 1 nên đến nhánh else, thực hiện step 10,11
  5. Step 10: $t2 = $t2 -1 = 0xffffffff
  6. Step 11: $t3 = $t3 + $t3 = 0x00000000

Asignment 2

- Khởi tạo:

```
.data
arr: .word 2019, 4588, 1, 2, 3, 5
.text
        add     $s1,$zero,-1    # Luu i = 0
        la      $s2,arr         # Dia chi A[0]
        addi    $s3,$zero,6     # Luu gia tri cua n
        addi    $s4,$zero,1     # Luu step
        lw      $s5,$zero,0     # Luu sum = A[0]
```

- Màn hình chạy:



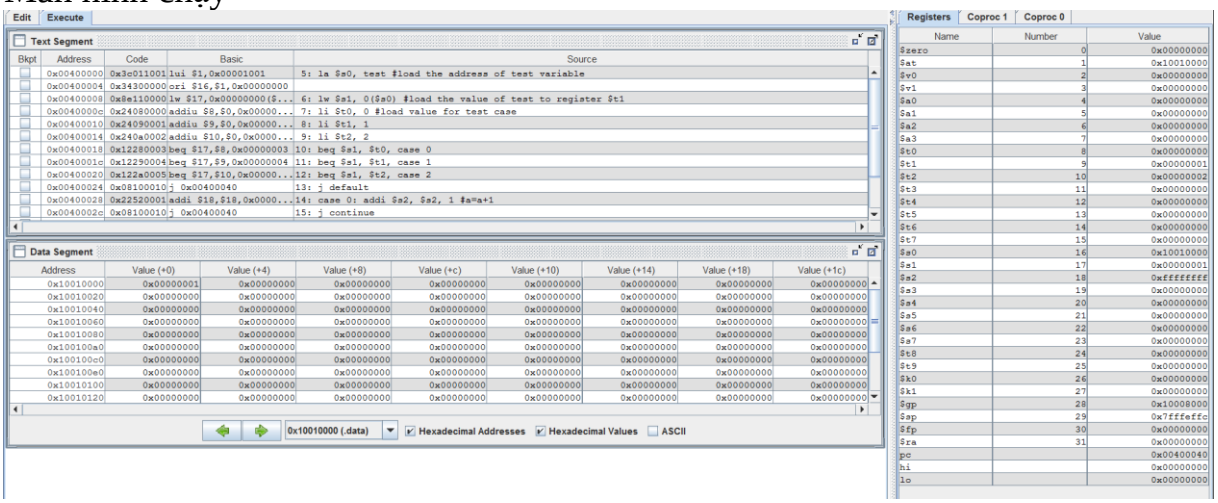
- Ý nghĩa:

loop: i=i+step;
Sum=sum+A[i];
If(I !=n) goto loop;

- Giải thích:
    1. Step 4,5,6,7,8: Khởi tạo
    2. Step 9: Bắt đầu loop: i = I +step = 0
    3. Step 10,11: t1 = 4*s1 (do 1 lệnh 4 bit)
    4. Step 12: Cập nhật A[i] vào biến tạm t1 (do t1 luôn được cập nhật + 4 mỗi lần loop nên A[i] luôn được cập nhật theo i)
    5. Step 13: load giá trị A[i] ($t1) vào biến tạm $t0
    6. Step 14: Thực hiện sum = sum + A[i] bằng lệnh add
    7. Step 15: Kiểm tra điều kiện nếu i ($s1) khác n ($s3) thì tiếp tục vòng lặp
- Kết quả: 0x00019DA => Chính xác

Assignment 3

- Màn hình chạy



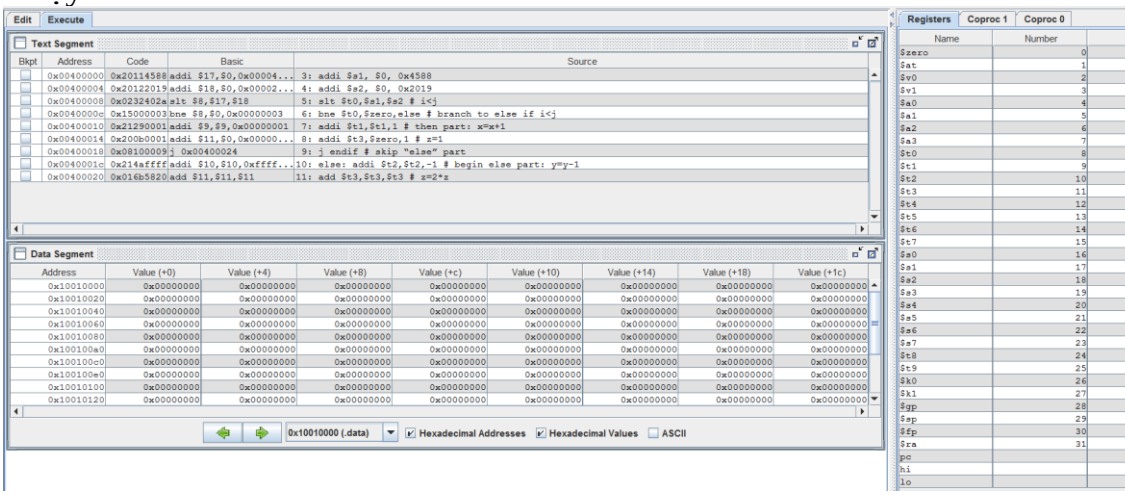
- Giải thích:
    1. Step 5: load test vào $s0
    2. Step 6: Gán $s1 = giá trị đầu của test = 1

3. Step 7,8,9: Gán 3 biến tạm $t0, $t1, $t2 các giá trị 0,1,2
4. Step 10,11,12: beq sẽ so sánh $s1 với 3 giá trị $t0, $t1, $t2. Nếu bằng thì đến các case tương ứng
5. Step 13: jump đến default
6. Step 14,15,16,16,17,19: So sánh các case để thực hiện các câu lệnh tương ứng

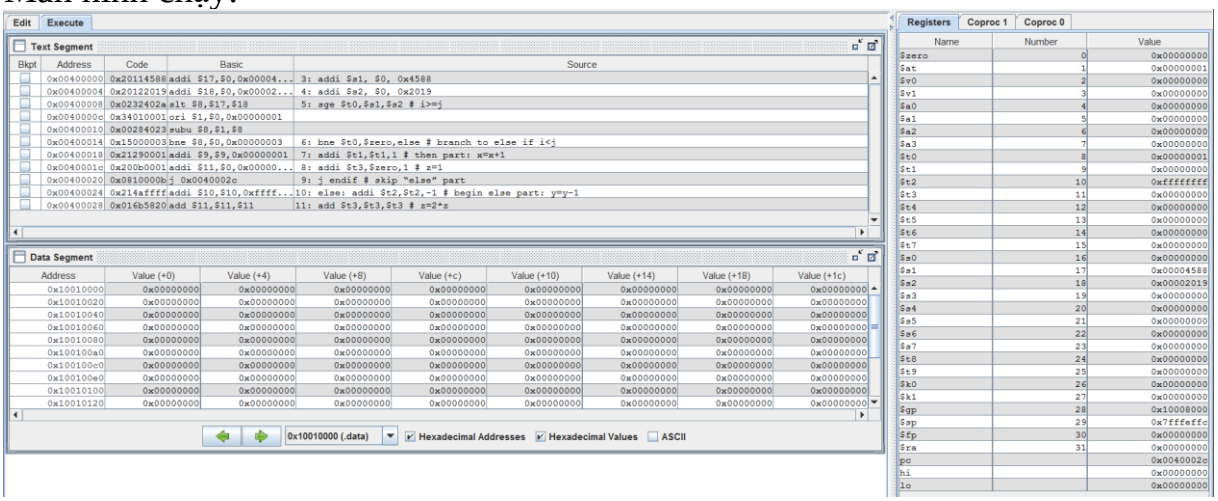
- Kết quả: test = 1 nên #s1 =St1 => case1=> $s2 = $s2 -$t1=> $s2 = 0xffffffff

Assignment 4

i<j

- Mản hình chạy:
  - 
- Giải thích:
  - Step 3,4: Khởi tạo $s1 = 0x4588, $s2 =0x2019
  - Step 5: slt so sánh nếu $s1 < $s2 thì $t0 = 1
  - Step 6: bne tạo nhánh nếu $t0 khác 0 thì đến nhánh else
  - Step 9:Do $t0 = 1 nên bỏ qua else, thực hiện step 7,8
  - Step 7: $t1 = $t1 +1 = 0x0x00001
  - Step 11: $t3 = 1

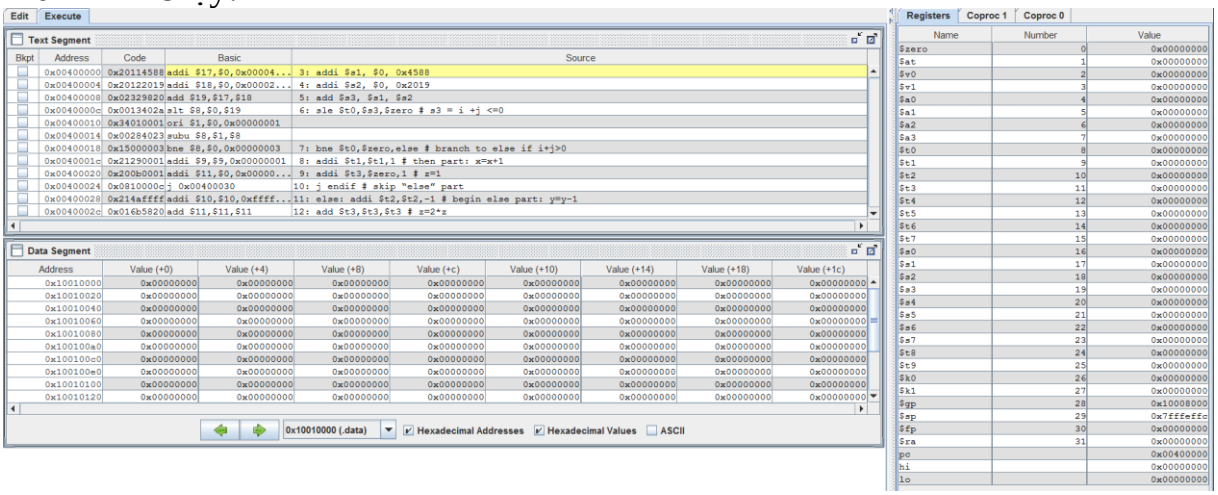
i >= j

- Màn hình chạy:
  - 

- Giải thích:
    1. Step 3,4: Khởi tạo $s1 = 0x4588, $s2 =0x2019
    2. Step 5: sge so sánh nếu $s1 >= $s2 thì $t0 = 1
    3. Step 6: bne tạo nhánh nếu $t0 khác 0 thì đến nhánh else
    4. Step 9:Do $t0 = 1 nên đến nhánh else, thực hiện step 10,11
    5. Step 10: $t2 = $t2 -1 = 0xffffffff
    6. Step 11: $t3 = $t3 + $t3 = 0x00000000

i+j <= 0

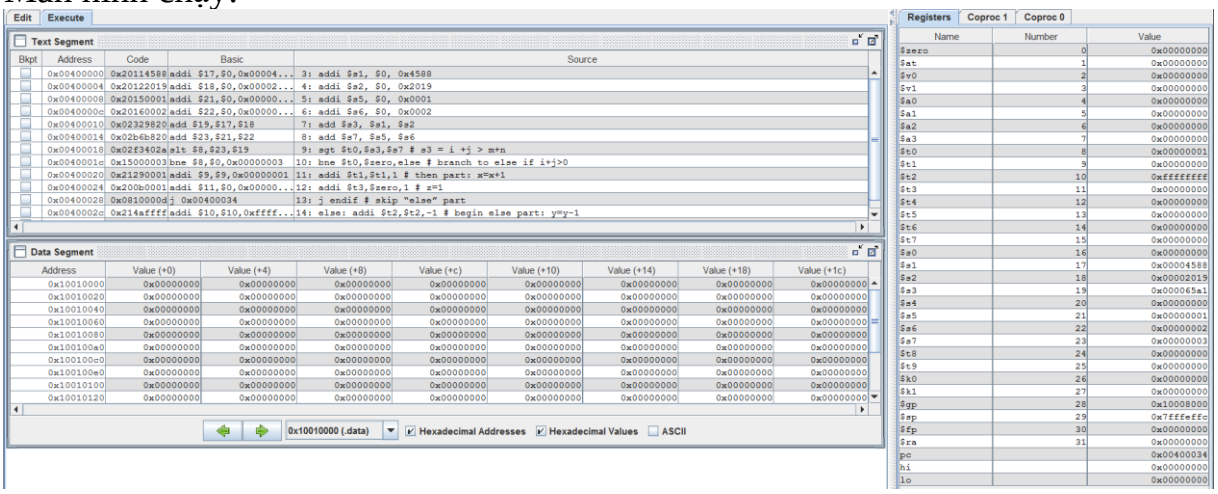
- Màn hình chạy:



- Giải thích:
    1. Step 3,4: Khởi tạo $s1 = 0x4588, $s2 =0x2019
    2. Step5: Gán s3 = s1+s2 = 0x00065a1
    3. Step 6: sle so sánh nếu $s3 <=0 thì $t0 = 1
    4. Step 6: bne tạo nhánh nếu $t0 khác 0 thì đến nhánh else
    5. Step 9:Do $t0 = 1 nên bỏ qua else, thực hiện step 7,8
    6. Step 7: $t1 = $t1 +1 = 0x0x00001
    7. Step 11: $t3 = 1

i+j>m+n

- Khởi tạo: m= 1, n=2
- Màn hình chạy:



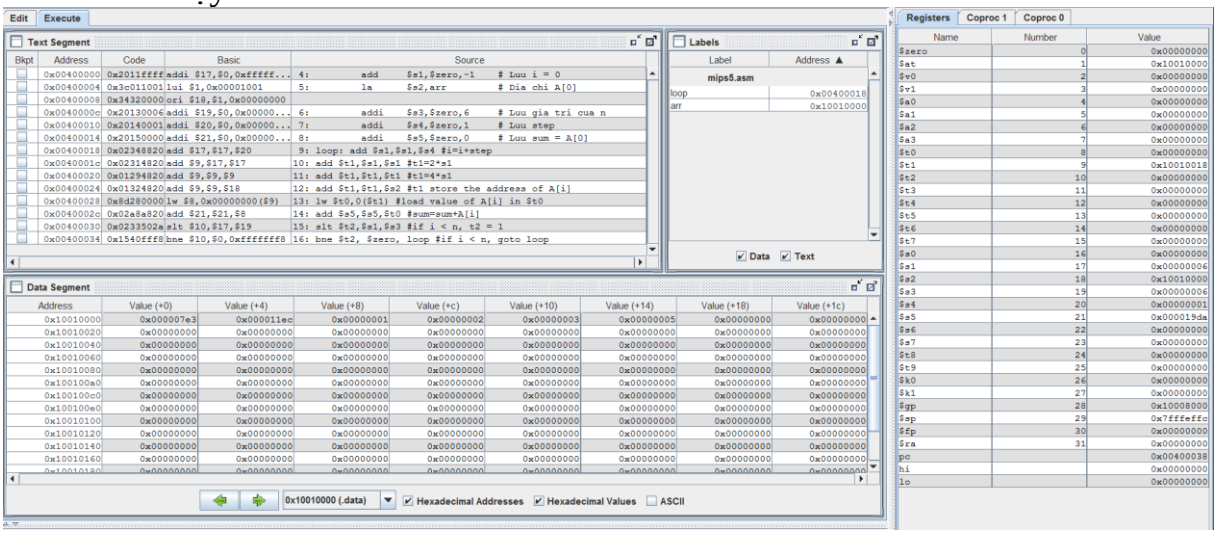
- Giải thích:

1. Step 3,4,5,6: Khởi tạo $s1 = 0x4588, $s2 =0x2019, $s5 = 0x0001, $s6 = 0x0002
2. Step 7,8: s3 = s1+s2= 2019+4588, s7 = s5+s6 = 1+2
3. Step 9: sgt so sánh nếu i+j >m+n thì $t0 = 1
4. Step 6: bne tạo nhánh nếu $t0 khác 0 thì đến nhánh else
5. Step 9:Do $t0 = 1 nên đến nhánh else, thực hiện step 10,11
6. Step 10: $t2 = $t2 -1 = 0xffffffff
7. Step 11: $t3 = $t3 + $t3 = 0x00000000

Assignment 5

i < n

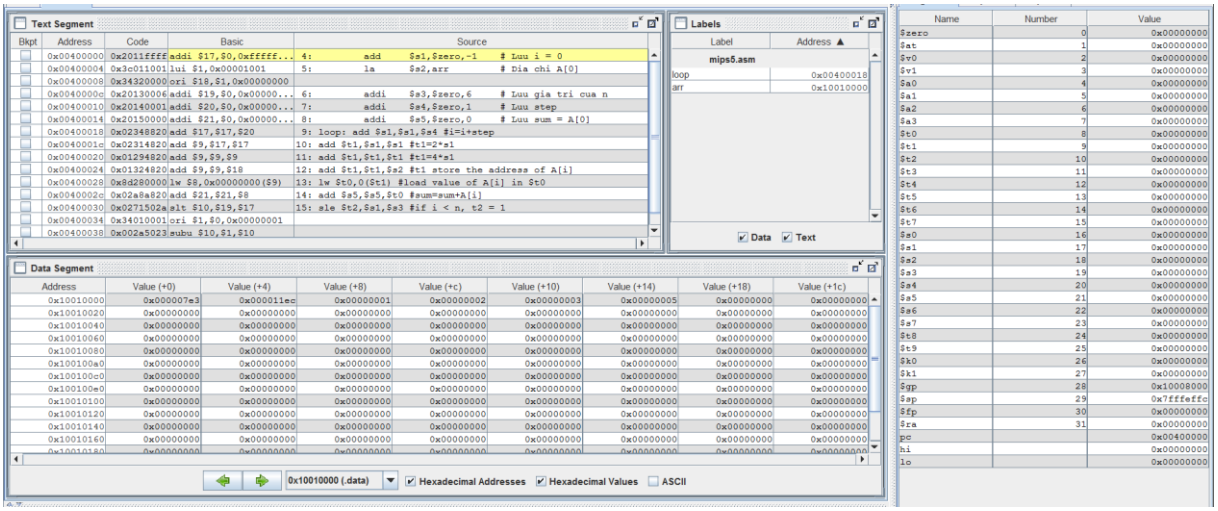
- Màn hình chạy:



- Giải thích:
    1. Step 4,5,6,7,8: Khởi tạo
    2. Step 9: Bắt đầu loop: i = I +step = 0
    3. Step 10,11: t1 = 4*s1 (do 1 lệnh 4 bit)
    4. Step 12: Cập nhật A[i] vào biến tạm t1 (do t1 luôn được cập nhật + 4 mỗi lần loop nên A[i] luôn được cập nhật theo i)
    5. Step 13: load giá trị A[i] ($t1) vào biến tạm $t0
    6. Step 14: Thực hiện sum = sum + A[i] bằng lệnh add
    7. Step 15: slt kiểm tra điều kiện I <n thì t2 = 1
    8. Step 16: bne kiểm tra nếu t2 khác 0 thì tiếp tục lặp
- Kết quả: 0x00019DA => Chính xác

i <= n

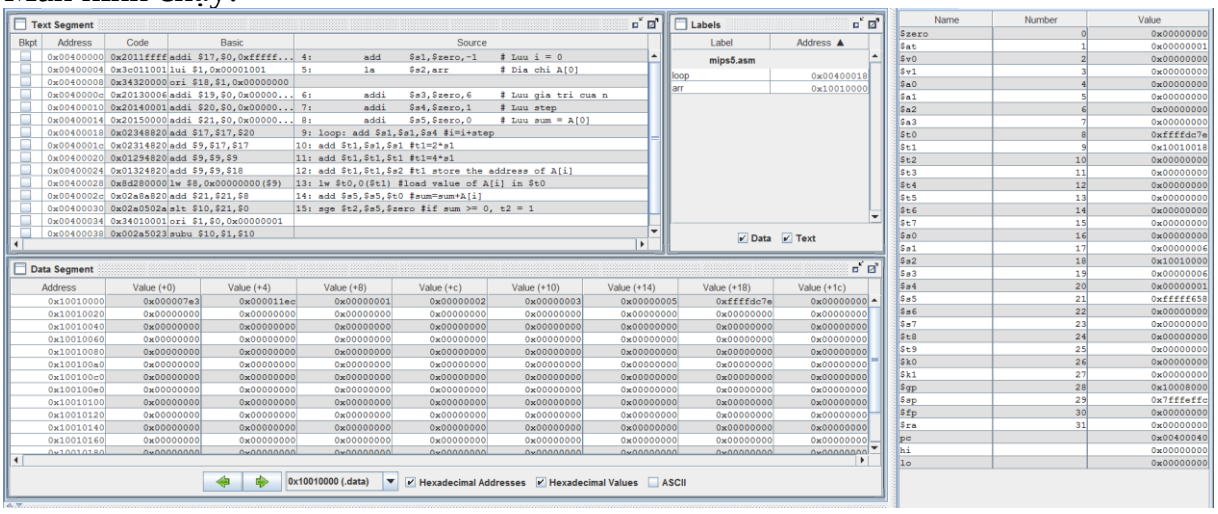
- Màn hình chạy:



- Giải thích:
    1. Step 4,5,6,7,8: Khởi tạo
    2. Step 9: Bắt đầu loop: i = I +step = 0
    3. Step 10,11: t1 = 4*s1 (do 1 lệnh 4 bit)
    4. Step 12: Cập nhật A[i] vào biến tạm t1 (do t1 luôn được cập nhật + 4 mỗi lần loop nên A[i] luôn được cập nhật theo i)
    5. Step 13: load giá trị A[i] ($t1) vào biến tạm $t0
    6. Step 14: Thực hiện sum = sum + A[i] bằng lệnh add
    7. Step 15: sle kiểm tra điều kiện I <=n thì t2 = 1
    8. Step 16: bne kiểm tra nếu t2 khác 0 thì tiếp tục lặp
- Kết quả: 0x00019DA => Chính xác

sum >= 0

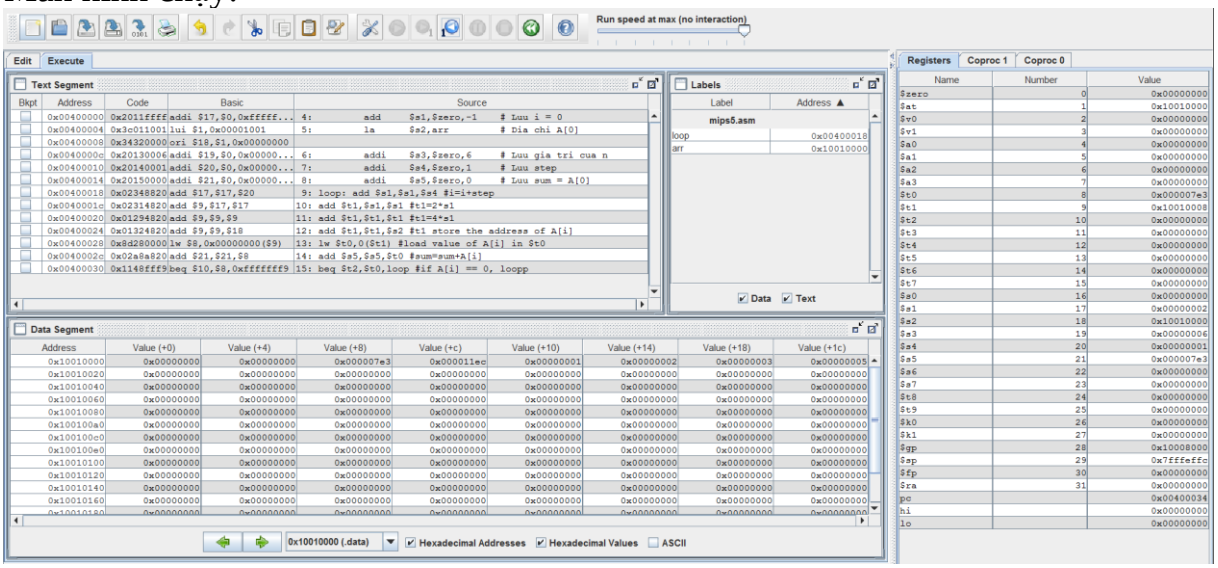
- Màn hình chạy:



- Giải thích:
    1. Step 4,5,6,7,8: Khởi tạo
    2. Step 9: Bắt đầu loop: i = I +step = 0
    3. Step 10,11: t1 = 4*s1 (do 1 lệnh 4 bit)
    4. Step 12: Cập nhật A[i] vào biến tạm t1 (do t1 luôn được cập nhật + 4 mỗi lần loop nên A[i] luôn được cập nhật theo i)
    5. Step 13: load giá trị A[i] ($t1) vào biến tạm $t0
    6. Step 14: Thực hiện sum = sum + A[i] bằng lệnh add

7. Step 15: sge kiểm tra điều kiện sum > = 0 thì t2 = 1
8. Step 16: bne kiểm tra nếu t2 khác 0 thì tiếp tục lặp

A[i] == 0

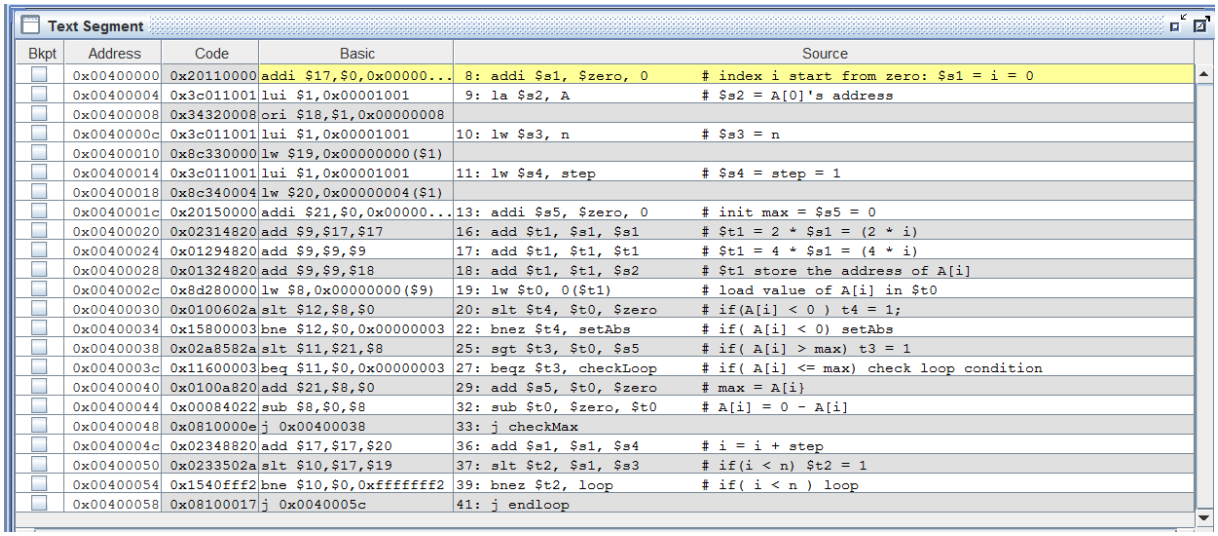
- Màn hình chạy:
- 
- Giải thích:
  1. Step 4,5,6,7,8: Khởi tạo
  2. Step 9: Bắt đầu loop: i = I +step = 0
  3. Step 10,11: t1 = 4*s1 (do 1 lệnh 4 bit)
  4. Step 12: Cập nhật A[i] vào biến tạm t1 (do t1 luôn được cập nhật + 4 mỗi lần loop nên A[i] luôn được cập nhật theo i)
  5. Step 13: load giá trị A[i] ($t1) vào biến tạm $t0
  6. Step 14: Thực hiện sum = sum + A[i] bằng lệnh add
  7. Step 15: beq kiểm tra điều kiện sum == 0 thì tiếp tục lặp

Assignment 6

- Khởi tạo:

```
n: .word 7
step: .word 1
A: .word 2019, 4588, 50, 1, 2, 3, -5, -6
```

- Màn hình chạy:



Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20110000	addi $17,$0,0x00000...	8: addi $s1, $zero, 0 # index i start from zero: $s1 = i = 0
	0x00400004	0x3c011001	lui $1,0x00001001	9: la $s2, A # $s2 = A[0]'s address
	0x00400008	0x34320008	ori $18,$1,0x00000008	
	0x0040000c	0x3c011001	lui $1,0x00001001	10: lw $s3, n # $s3 = n
	0x00400010	0x8c330000	lw $19,0x00000000($1)	
	0x00400014	0x3c011001	lui $1,0x00001001	11: lw $s4, step # $s4 = step = 1
	0x00400018	0x8c340004	lw $20,0x00000004($1)	
	0x0040001c	0x20150000	addi $21,$0,0x00000...	13: addi $s5, $zero, 0 # init max = $s5 = 0
	0x00400020	0x02314820	add $9,$17,$17	16: add $t1, $s1, $s1 # $t1 = 2 * $s1 = (2 * i)
	0x00400024	0x01294820	add $9,$9,$9	17: add $t1, $t1, $t1 # $t1 = 4 * $s1 = (4 * i)
	0x00400028	0x01324820	add $9,$9,$18	18: add $t1, $t1, $s2 # $t1 store the address of A[i]
	0x0040002c	0x8d280000	lw $8,0x00000000($9)	19: lw $t0, 0($t1) # load value of A[i] in $t0
	0x00400030	0x0100602a	slt $12,$8,$0	20: slt $t4, $t0, $zero # if(A[i] < 0) t4 = 1;
	0x00400034	0x15800003	bne $12,$0,0x00000003	22: bnez $t4, setAbs # if(A[i] < 0) setAbs
	0x00400038	0x02a8582a	slt $11,$21,$8	25: sgt $t3, $t0, $s5 # if(A[i] > max) t3 = 1
	0x0040003c	0x11600003	beq $11,$0,0x00000003	27: beqz $t3, checkLoop # if(A[i] <= max) check loop condition
	0x00400040	0x0100a820	add $21,$8,$0	29: add $s5, $t0, $zero # max = A[i}
	0x00400044	0x00084022	sub $8,$0,$8	32: sub $t0, $zero, $t0 # A[i] = 0 - A[i]
	0x00400048	0x0810000e	j 0x00400038	33: j checkMax
	0x0040004c	0x02348820	add $17,$17,$20	36: add $s1, $s1, $s4 # i = i + step
	0x00400050	0x0233502a	slt $10,$17,$19	37: slt $t2, $s1, $s3 # if(i < n) $t2 = 1
	0x00400054	0x1540fff2	bne $10,$0,0xfffffff2	39: bnez $t2, loop # if(i < n) loop
	0x00400058	0x08100017	j 0x0040005c	41: j endloop

- Giải thích:
    1. Step 8, 9, 10, 11, 13: Khởi tạo $s1 = i=0, $s2 = A, $s3 = n, $s4 = step = 1, $s4 = max = 0
    2. Step 16,17: cập nhật $t1 = 4.$s1 (vì để nhảy sang A[i] tiếp theo cần thêm 4 bit)
    3. Step 18, 19: Cập nhật địa chỉ A[i] cho biến tạm $t0
    4. Step 20,22: Nếu A[i] < 0 thì cập nhật trị tuyệt đối cho A[i]
    5. Step 25: sqt check điều kiện Nếu A[i] > max thì cho t3 = 1
    6. Step 27: beqz Kiểm tra t3, nếu t3 = 0 (nghĩa là A[i] <= max) thì tiếp tục vòng lặp
    7. Step 29: Nếu A[i] > max thì cập nhật A[i] cho max ($s4)
    8. Step 32, 33: Biểu thị khi gặp số âm sẽ cập nhật trị tuyệt đối và quay lại kiểm tra max
    9. Step 36,37,39: Kiểm tra điều kiện vòng lặp
- Kết quả: $s5 = 0x000011ec = 4588
  ⇨ Kết quả chính xác