MIPS architecture

Lab3, C problem

find the smallest odd number

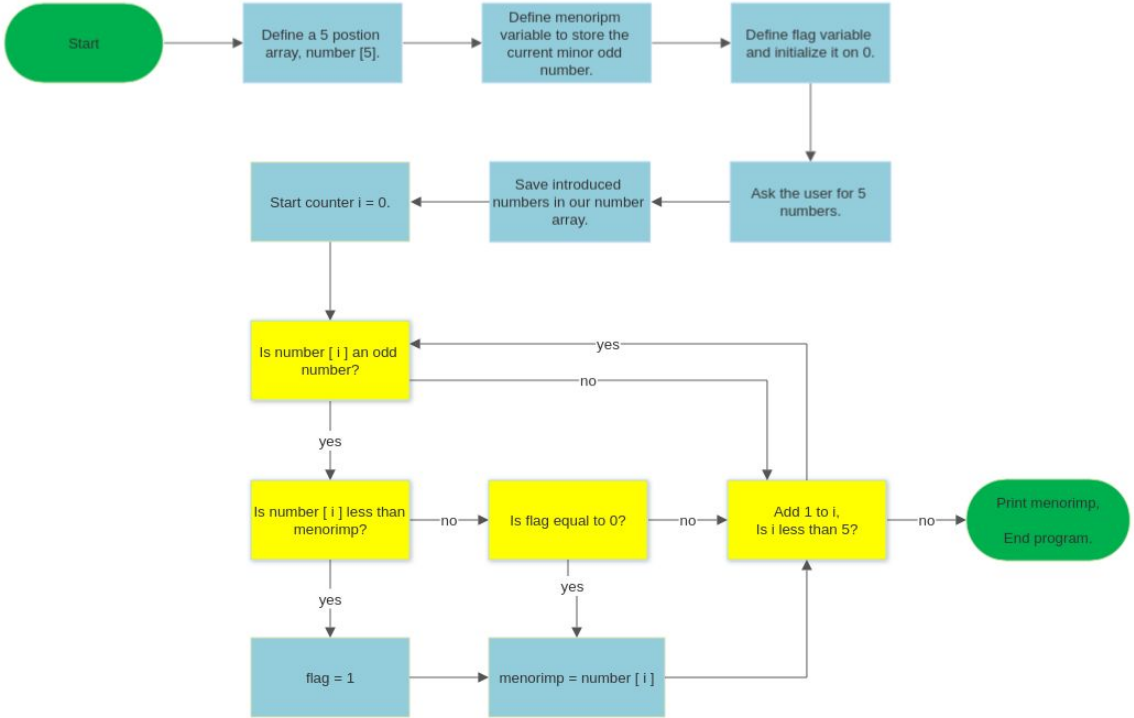
Team 4:

José Alberto Gómez Díaz
Ricardo Bazaldúa Calderón
Perla Noemí Méndez Zavaleta
Christian Aaron Ortega Blanco

Problem description

A. Given the numbers f,g,h,i,j and assuming that at least one of them is an odd number, make a C program that decides the smallest odd number.
B. Translate from C program to MIPS assembler.
C. Translate from assembler to machine code.

Flux diagram



C Program

```c
#include <stdio.h>
#include <conio.h>
int main() {
    int number[5];
    int menorimp;
    int flag=0;
    int modulo=0;
    printf("F ");
    scanf("%d", &number[0]);
    printf("G ");
    scanf("%d", &number[1]);
    printf("H ");
    scanf("%d", &number[2]);
    printf("I ");
    scanf("%d", &number[3]);
    printf("J ");
    scanf("%d", &number[4]);
    // displays output

for(int i=0;i<5;i++){
        modulo=number[i]/2;
        modulo=modulo*2;
        modulo=number[i]-modulo;

        if ( modulo == 1) {

            if(!flag){
                menorimp=number[i];
                flag=1;
            }
            else if(number[i]<menorimp) menorimp=number[i];


        }
}
    printf("the minimum odd number is: %d", menorimp);
    return 0;
}
```

Simulation:

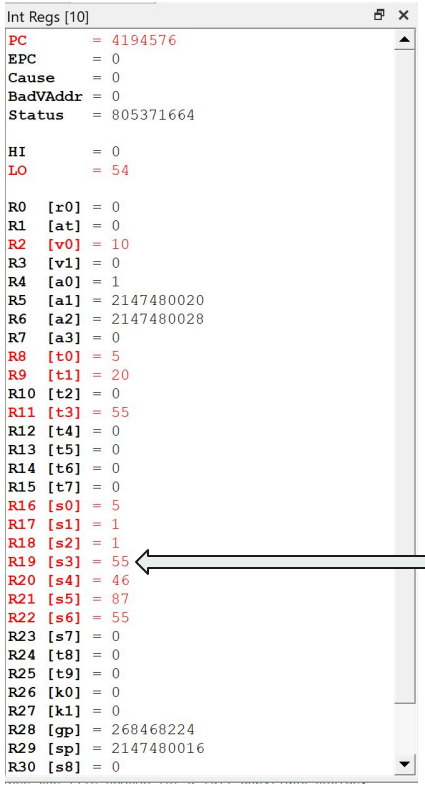
```
F 68
G 96
H 46
I 87
J 55
the minimum odd number is: 55
--------------------------------
Process exited after 10.92 seconds with return value 0
Presione una tecla para continuar . . .
```

Assembler MIPS

```
15  main:
16
17      addi $s1,$0,0  #module
18      addi $s0,$0,0  #i
19      addi $t0,$0,5  #t0=5
20
21
22      addi $s2, $zero, 68
23      addi $s3, $zero, 96
24      addi $s4, $zero, 46
25      addi $s5, $zero, 87
26      addi $s6, $zero, 55
27  #index = $t1 array
28          addi $t1, $zero, 0
29      sw $s2, number($t1)
30          addi $t1, $t1, 4
31      sw $s3, number($t1)
32          addi $t1, $t1, 4
33      sw $s4, number($t1)
34          addi $t1, $t1, 4
35      sw $s5, number($t1)
36          addi $t1, $t1, 4
37      sw $s6, number($t1)
38
39
40      addi $t1, $zero, 0 #clear $t1 to 0
41
42      addi $s2, $zero, 0  # flag
43      addi $s3, $zero, 0  # menorimp
```

```
44
45      loop: bne $s0,$t0,for #if i !=5 salta a for
46          j end
47      for:
48          #finding module  1 odd, 0 even
49          lw $t3, number($t1)
50          div $s1,$t3,2
51          mul $s1,$s1,2
52          sub $s1, $t3, $s1  #$s1 module
53
54          bne $s1, 1, else  #if module !=1 go else
55              bgtz $s2,findless #if flag > 0,
56                  move $s3, $t3 #set the first odd number
57                  addi $s2, $zero, 1 #close the funtion
58              findless:
59                  bgt $t3,$s3,else #compare the previous odd to find the smallest
60                  move $s3, $t3 #save the smallest odd number
61      else:
62
63
64          addi $t1, $t1, 4 # set the next position on array
65          addi $s0, $s0,1 #i=i+1
66
67          j loop
68      end:
```

MIPS results



smallest odd number

number[5]= {68,96,46,87,55}

R=$S3

$s3=55d

Machine code

```
[00400038] 20110000  addi $17, $0, 0        ; 17: addi $s1,$0,0 #module
[0040003c] 20100000  addi $16, $0, 0        ; 18: addi $s0,$0,0 #i
[00400040] 20080005  addi $8, $0, 5         ; 19: addi $t0,$0,5 #t0=5
[00400044] 20120044  addi $18, $0, 68       ; 22: addi $s2, $zero, 68
[00400048] 20130060  addi $19, $0, 96       ; 23: addi $s3, $zero, 96
[0040004c] 2014002e  addi $20, $0, 46       ; 24: addi $s4, $zero, 46
[00400050] 20150057  addi $21, $0, 87       ; 25: addi $s5, $zero, 87
[00400054] 20160037  addi $22, $0, 55       ; 26: addi $s6, $zero, 55
[00400058] 20090000  addi $9, $0, 0         ; 28: addi $t1, $zero, 0
[0040005c] 3c010040  lui $1, 64           ; 29: sw $s2, number($t1)
[00400060] 00290821  addu $1, $1, $9
[00400064] ac320024  sw $18, 36($1)
[00400068] 21290004  addi $9, $9, 4         ; 30: addi $t1, $t1, 4
[0040006c] 3c010040  lui $1, 64           ; 31: sw $s3, number($t1)
[00400070] 00290821  addu $1, $1, $9
[00400074] ac330024  sw $19, 36($1)
[00400078] 21290004  addi $9, $9, 4         ; 32: addi $t1, $t1, 4
[0040007c] 3c010040  lui $1, 64           ; 33: sw $s4, number($t1)
[00400080] 00290821  addu $1, $1, $9
[00400084] ac340024  sw $20, 36($1)
[00400088] 21290004  addi $9, $9, 4         ; 34: addi $t1, $t1, 4
[0040008c] 3c010040  lui $1, 64           ; 35: sw $s5, number($t1)
[00400090] 00290821  addu $1, $1, $9
[00400094] ac350024  sw $21, 36($1)
[00400098] 21290004  addi $9, $9, 4         ; 36: addi $t1, $t1, 4
[0040009c] 3c010040  lui $1, 64           ; 37: sw $s6, number($t1)
[004000a0] 00290821  addu $1, $1, $9
[004000a4] ac360024  sw $22, 36($1)
[004000a8] 20090000  addi $9, $0, 0         ; 40: addi $t1, $zero, 0 #clear $t1 to 0
[004000ac] 20120000  addi $18, $0, 0        ; 42: addi $s2, $zero, 0 # flag
[004000b0] 20130000  addi $19, $0, 0        ; 43: addi $s3, $zero, 0 # menorimp
```

```
[004000b4] 16080002  bne $16, $8, 8 [for-0x004000b4]; 45: bne $s0,$t0,for #if i !=5 salta a for
[004000b8] 08100043  j 0x0040010c [end]       ; 46: j end
[004000bc] 3c010040  lui $1, 64             ; 49: lw $t3, number($t1)
[004000c0] 00290821  addu $1, $1, $9
[004000c4] 8c2b0024  lw $11, 36($1)
[004000c8] 34010002  ori $1, $0, 2          ; 50: div $s1,$t3,2
[004000cc] 0161001a  div $11, $1
[004000d0] 00008812  mflo $17
[004000d4] 34010002  ori $1, $0, 2          ; 51: mul $s1,$s1,2
[004000d8] 72218802  mul $17, $17, $1
[004000dc] 01718822  sub $17, $11, $17      ; 52: sub $s1, $t3, $s1 #$s1 module
[004000e0] 34010001  ori $1, $0, 1          ; 54: bne $s1, 1, else #if module !=1 go else
[004000e4] 14310007  bne $1, $17, 28 [else-0x004000e4]
[004000e8] 1e400003  bgtz $18 12 [findless-0x004000e8]
[004000ec] 000b9821  addu $19, $0, $11      ; 56: move $s3, $t3 #set the first odd number
[004000f0] 20120001  addi $18, $0, 1        ; 57: addi $s2, $zero, 1 #close the funtion
[004000f4] 026b082a  slt $1, $19, $11       ; 59: bgt $t3,$s3,else #compare the previous odd to
find the smallest
[004000f8] 14200002  bne $1, $0, 8 [else-0x004000f8]
[004000fc] 000b9821  addu $19, $0, $11      ; 60: move $s3, $t3 #save the smallest odd number
[00400100] 21290004  addi $9, $9, 4         ; 64: addi $t1, $t1, 4 # set the next position on array
[00400104] 22100001  addi $16, $16, 1       ; 65: addi $s0, $s0,1 #i=i+1
[00400108] 0810002d  j 0x004000b4 [loop]    ; 67: j loop
[0040010c] 3402000a  ori $2, $0, 10         ; 71: li $v0, 10 # Sets $v0 to "10" to select exit syscall
[00400110] 0000000c  syscall              ; 73: syscall # Exit
```

Machine code

ADDI $s1 $zero 0x0000

addi rt, rs, imm

001000 10001 00000 00000

Name	Register Number
$t0-$t7	8-15
$s0-$s7	16-23

I Type			
op	rs	rt	imm
001000	00000	10001	0000000000000000

Machine code

```
00100000000010001000000000000000
00100000000010000000000000000000
00100000000010000000000000000101
00100000000010010000000000000110
00100000000010011000000000000111
00100000000010100000000000001000
00100000000010101000000000000101
00100000000010110000000000000010
00100000000010010000000000000000
00111100000000001000000001000000
00000000001010010000100000100001
10101100001100100000000000100100
00100001001010010000000000000100
00111100000000001000000001000000
00000000001010010000100000100001
10101100001100110000000000100100
00100001001010010000000000000100
00111100000000001000000001000000
00000000001010010000100000100001
10101100001101000000000000100100
00100001001010010000000000000100
00111100000000001000000001000000
00000000001010010000100000100001
10101100001101010000000000100100
00100001001010010000000000000100
00111100000000001000000001000000
00000000001010010000100000100001
10101100001101100000000000100100
```

```
00100000000010010000000000000000
00100000000010010000000000000000
00100000000010011000000000000000
00010110000010000000000000000010
00001000000100000000000001000011
00111100000000010000000001000000
00000000001010010000100000100001
10001100001010110000000000100100
00110100000000001000000000000010
00000001011000010000000000011010
00000000000000001000100000010010
00110100000000001000000000000010
01110000010110001000000000000010
00000001011100011000100000100010
00110100000000001000000000000001
00010100001100010000000000000111
00011110010000000000000000000011
00000000000010111001100000100001
00100000001001000000000000000001
00000010011010110000100000101010
00010100001000000000000000000010
00000000000010111001100000100001
00100001001010010000000000000100
00100010000100000000000000000001
00001000000100000000000000101101
00110100000000100000000000001010
00000000000000000000000000001100
```

Conclusions

As shown on the simulation results, the behavior of both programs is pretty much the same. visualizing that the assembler code as a different structure, the C program had to pass through different ways of thinking to ensure that the result is going to be almost directly converted to MIPS. contemplating that, the assembler programing was very step forward.