# Roulette

Project 1

Lei wan

10-31-2014

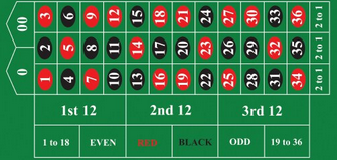
Content

1. Introduce
   1. Rules
2. Development
   1. Difficult points
   2. Call function
   3. Game function
3. Pseudo Code
4. Flowchart
5. Code

**1. Introduction**

**1.1 Rules**

Roulette is a standard game in Casino. Player have lots of choices to bet. Players can bet on Single number, Column, Half row, Single row, Color, Even/Odd, etc. Of cause players can bet combination.



Roulette table picture

**2. Development**

**2.1 Difficult points**

In this game, there are plenty of variable, but I can only use just 14 registers. Surely I can use stack, yet I have to call many function, and the stack I plan to store the lr.

**2.2 Call Function**

In this game, the ball is random to stay on any number (0-36) on the roulette, so I need a random number. I call C function time(), srand(), rand(); I input and output information I need call C function printf(),scanf(); when input is type string, I call C function strcmp().

Otherwise, I design 5 way to make bet on Single number, Column, Single row, Color and Odd/Even. So I get function gamesinglenum(), gamecolumn(), gamesingler(), gamecolor() and gameoe().

I combinate divede and rand() function into one function to realize the random number. It is function random()

**2.3 Game function**

1. Player have an account $100

2. The main menu. Player could choose exit or different way to bet. Player also can combine up to five ways on one game. E.g. player could choose bet 1.Single number and 2. Column or choose all five ways. Once you choose one way, this way will disappear from the menu. E.g. if you choose 4 .Color, the 4 option will disappear from the menu, so one game you can not choose same option twice.

3. When the game end, AI will output the ball stay which number; the player is win or

Lose; the balance will update on player’s account.

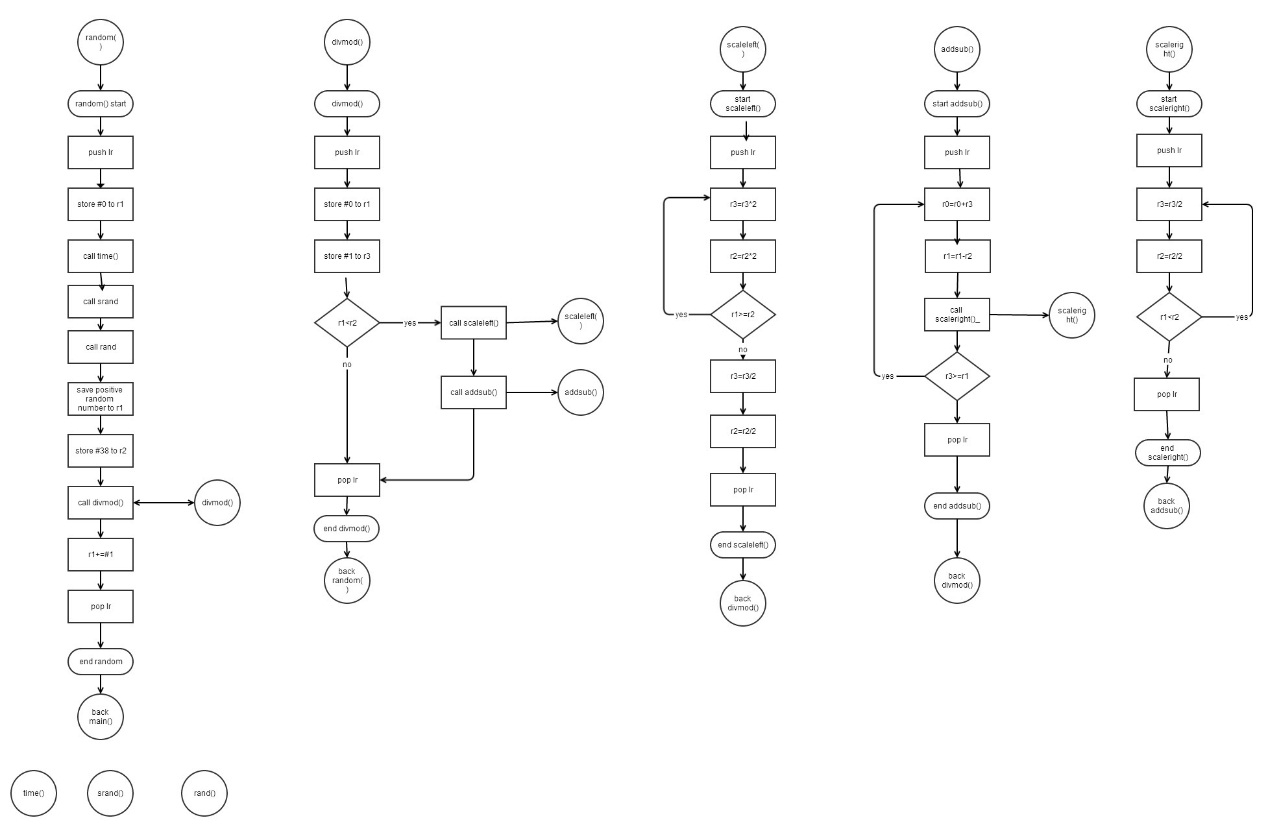
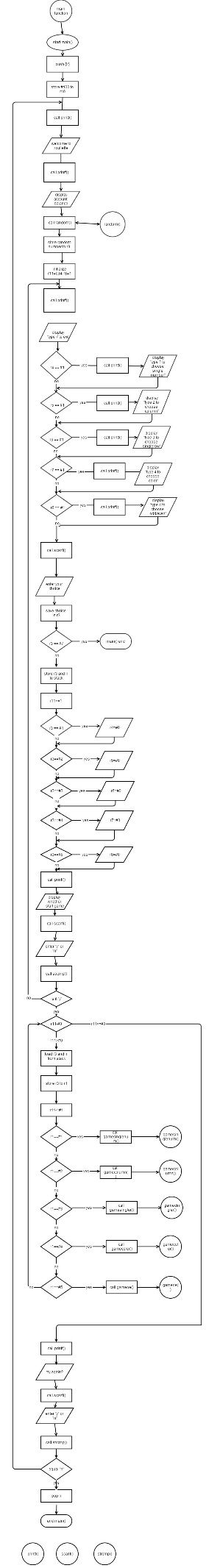
1. AI will ask player try again or not.

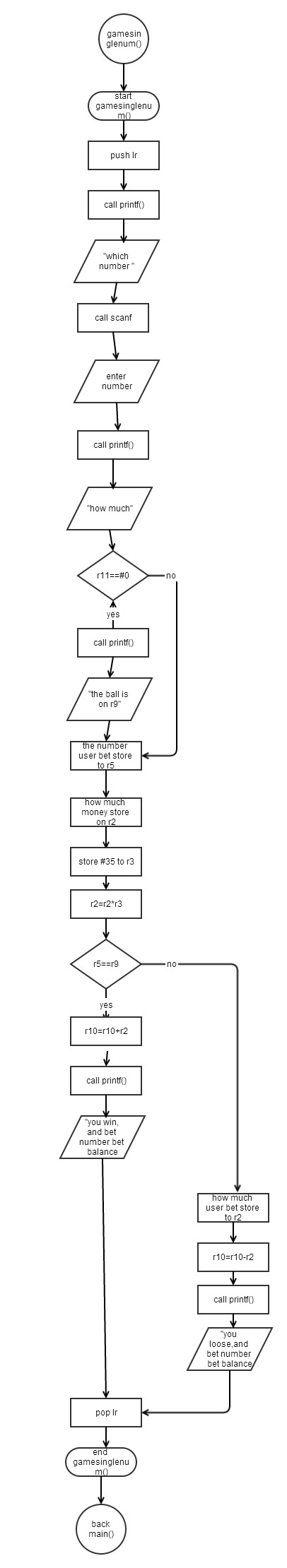
**3.Pseudo Code**

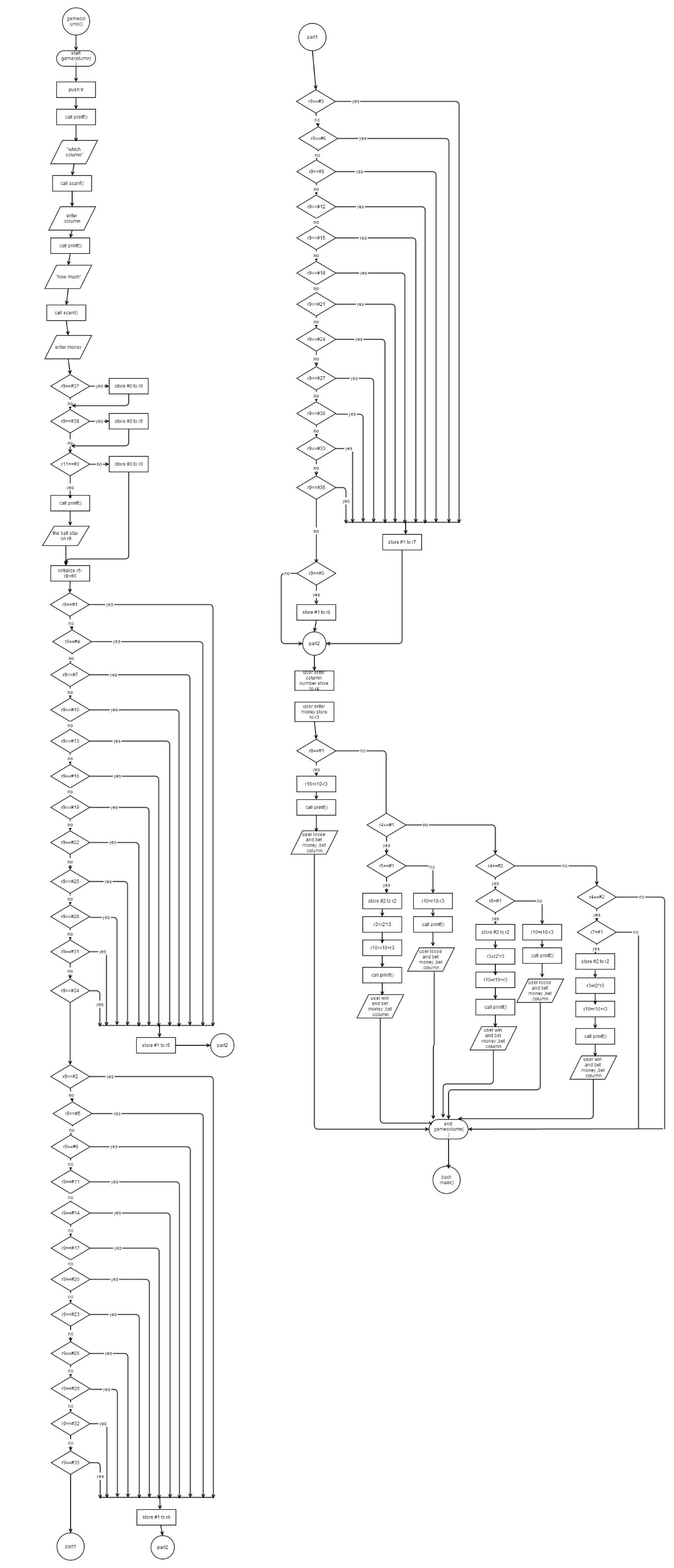
|  |
| --- |
|  |
|  |  |
|  |  |
|  |  |
|  | **ldr** r3,addr\_store1 |
|  | **ldr** r3,[r3] |
|  |  |
|  | **cmp** r3,*#7* |
|  | **beq** final |
|  | **push** {r3,lr} |
|  | **add** r11,r11,*#1* |
|  | **cmp** r3,*#1* |
|  | **beq** jump1 |
|  | **bne** jump2 |
|  | jump1: |
|  | **mov** r4,*#0* |
|  |  |
|  | jump2: |
|  | **cmp** r3,*#2* |
|  | **beq** jump3 |
|  | **bne** jump4 |
|  |  |
|  | jump3: |
|  | **mov** r5,*#0* |
|  |  |
|  | jump4: |
|  | **cmp** r3,*#3* |
|  | **beq** jump5 |
|  | **bne** jump6 |
|  |  |
|  | jump5: |
|  | **mov** r6,*#0* |
|  | jump6: |
|  | **cmp** r3,*#4* |
|  | **beq** jump7 |
|  | **bne** jump8 |
|  | jump7: |
|  | **mov** r7,*#0* |
|  |  |
|  | jump8: |
|  | **cmp** r3,*#5* |
|  | **beq** jump9 |
|  | **bne** jump10 |
|  |  |
|  | jump9: |
|  | **mov** r8,*#0* |
|  |  |
|  | jump10: |
|  |  |
|  | **ldr** r0,addr\_in6 |
|  | **bl** printf |
|  |  |
|  |  |
|  | **ldr** r0,addr\_format1 |
|  | **ldr** r1,addr\_store2 |
|  | **bl** scanf |
|  |  |
|  | **ldr** r1,addr\_store2 |
|  | **ldr** r0,addr\_cmp1 |
|  | **bl** strcmp |
|  | **beq** goto |
|  | **bne** game |
|  |  |
|  |  |
|  | goto: |
|  | **cmp** r11,*#0* |
|  | **ble** end @jump out no more bet |
|  | **pop** {r3,lr} |
|  | **mov** r1,r3 |
|  | **sub** r11,r11,*#1* |
|  | **cmp** r1,*#1* |
|  | **bleq** gamesinglenum |
|  | **cmp** r1,*#2* |
|  | **bleq** gamecolumn |
|  | **cmp** r1,*#3* |
|  | **bleq** gamesingler |
|  | **cmp** r1,*#4* |
|  | **bleq** gamecolor |
|  | **cmp** r1,*#5* |
|  | **bleq** gameoe |
|  | **b** goto |

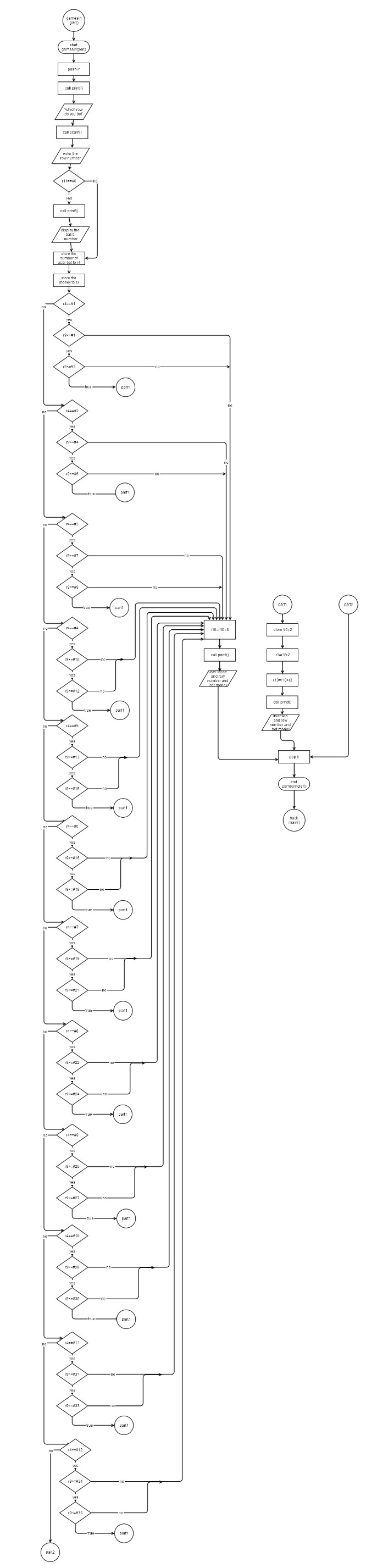
In order to realize the combination of choices. I use r11 to calculate the efficient choice. The efficient choice here is from 1 to 5.(Single number, Column, single row, color and odd/even). Every time choose the 5 option [r11]+1(r11 initialize #0) and the same time push the choice to stack. When choice finished pop stack, by comparing the pop value with the function value to choose call which function.

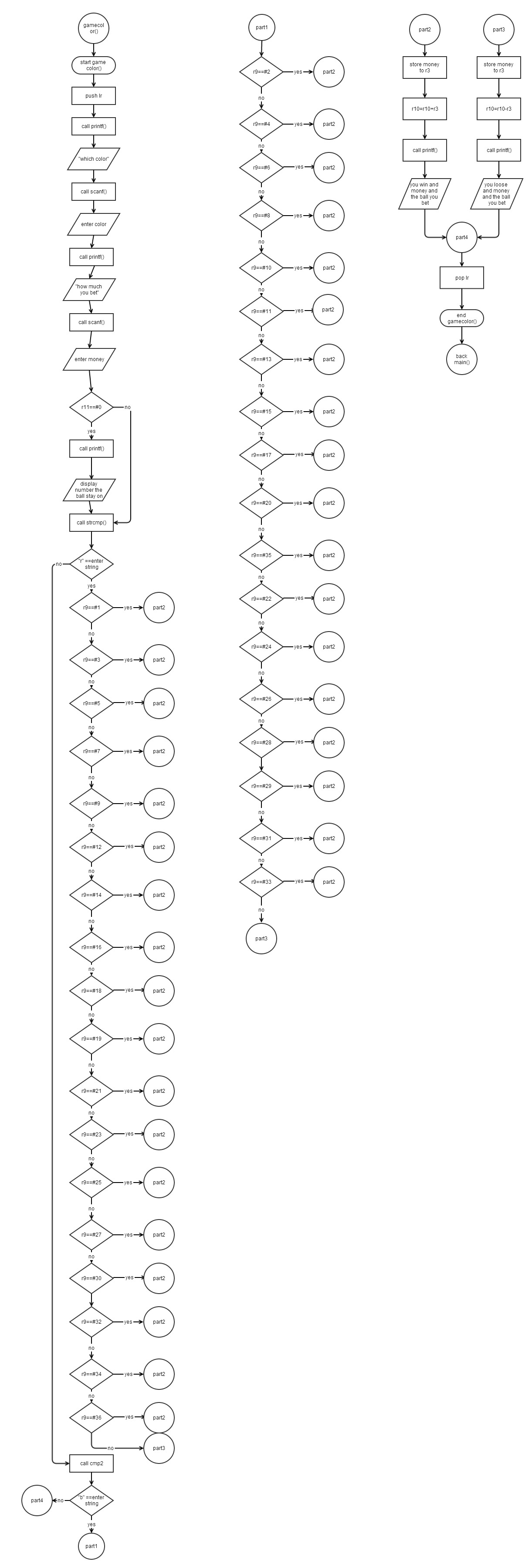
**4.Flowchart**

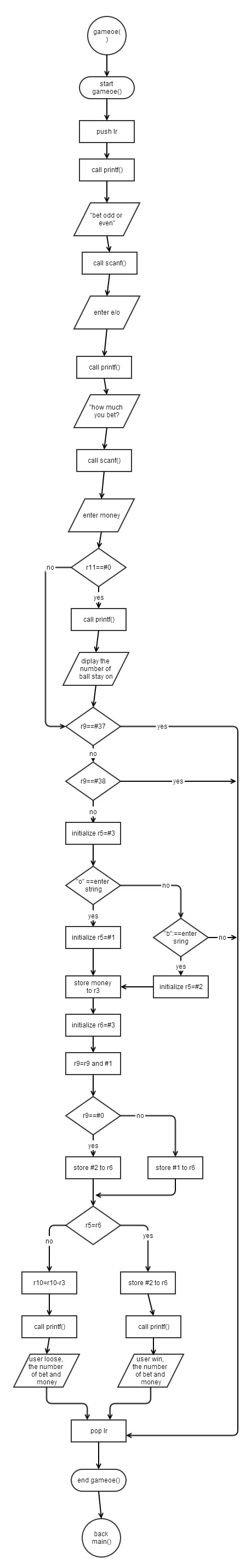












**Main()**

.data

in1: .asciz "type 1 to choose Single Number\n"

in2:.asciz"type 2 to choose Column\n"

in3:.asciz"type 3 to choose Single Row\n"

in4:.asciz"type 4 to choose Red/Black\n"

in5:.asciz"type 5 to choose Even/Odd\n"

in6:.asciz"Do you finish the choice and to start the game? (y/n)\n"

in7:.asciz"Do you wanna try again? (y/n)\n"

in8:.asciz"\nYour account balance is: %d\n"

in9:.asciz"\n Welcome to Roullette Game\n"

in10:.asciz"type 7 to exit\n"

format:.asciz"%d"

format1:.asciz"%s"

.balign 4

store1:.word 0

.balign 4

store2:.word 0

.balign 4

store3:.word 0

.balign 4

store4:.word 0 @exit lr value

cmp:.asciz"n"

cmp1:.asciz"y"

.text

.global main

main:

push {lr}

mov r10, #100 @acount balance initiate $100

more:

ldr r0,addr\_in9

bl printf

ldr r0,addr\_in8

mov r1,r10

bl printf

bl random

mov r9,r1 @recorder the ball

mov r11,#0 @counter

mov r4,#1

mov r5,#1

mov r6,#1

mov r7,#1

mov r8,#1 @judge whether keep on the same method

game:

ldr r0,addr\_in10

bl printf

cmp r4,#1

beq singlenum

bne run1 @~~~~~~~~~~~~~

singlenum:

ldr r0, addr\_in1

bl printf

run1:

cmp r5,#1

beq column

bne run2

column:

ldr r0, addr\_in2

bl printf

run2:

cmp r6,#1

beq singler

bne run3

singler:

ldr r0,addr\_in3

bl printf

run3:

cmp r7,#1

beq color

bne run4

color:

ldr r0,addr\_in4

bl printf

run4:

cmp r8,#1

beq oe

bne run5

oe:

ldr r0, addr\_in5

bl printf

run5:

ldr r0,addr\_format

ldr r1,addr\_store1 @number of your choice

bl scanf

ldr r3,addr\_store1

ldr r3,[r3]

cmp r3,#7

beq final

push {r3,lr}

add r11,r11,#1

cmp r3,#1

beq jump1

bne jump2

jump1:

mov r4,#0

jump2:

cmp r3,#2

beq jump3

bne jump4

jump3:

mov r5,#0

jump4:

cmp r3,#3

beq jump5

bne jump6

jump5:

mov r6,#0

jump6:

cmp r3,#4

beq jump7

bne jump8

jump7:

mov r7,#0

jump8:

cmp r3,#5

beq jump9

bne jump10

jump9:

mov r8,#0

jump10:

ldr r0,addr\_in6

bl printf

ldr r0,addr\_format1

ldr r1,addr\_store2

bl scanf

ldr r1,addr\_store2

ldr r0,addr\_cmp1

bl strcmp

beq goto

bne game

goto:

cmp r11,#0

ble end @jump out no more bet

pop {r3,lr}

mov r1,r3

sub r11,r11,#1

cmp r1,#1

bleq gamesinglenum

cmp r1,#2

bleq gamecolumn

cmp r1,#3

bleq gamesingler

cmp r1,#4

bleq gamecolor

cmp r1,#5

bleq gameoe

b goto

end:

ldr r0,addr\_in7

bl printf

ldr r0,addr\_format1

ldr r1,addr\_store3

bl scanf

ldr r0, addr\_cmp

ldr r1,addr\_store3

bl strcmp

beq final

bne more

final:

pop {lr}

bx lr

addr\_in1:.word in1

addr\_in2:.word in2

addr\_in3:.word in3

addr\_in4:.word in4

addr\_in5:.word in5

addr\_in6:.word in6

addr\_in7:.word in7

addr\_in8:.word in8

addr\_in9:.word in9

addr\_in10:.word in10

addr\_store1:.word store1

addr\_store2:.word store2

addr\_store3:.word store3

addr\_format:.word format

addr\_format1:.word format1

addr\_cmp:.word cmp

addr\_cmp1:.word cmp1

.global printf

.global strcmp

.global srand

**Random()**

.data

.text

.global scaleright

scaleRight:

push {lr}

doWhile\_r1\_lt\_r2:

mov r3,r3,ASR #1

mov r2,r2,ASR #1

cmp r1,r2

blt doWhile\_r1\_lt\_r2

pop {lr}

bx lr

.global addsub

addsub:

push {lr}

doWhile\_r3\_ge\_1:

add r0,r0,r3

sub r1,r1,r2

bl scaleRight

cmp r3,#1

bge doWhile\_r3\_ge\_1

pop {lr}

bx lr

.global scaleleft

scaleleft:

push {lr}

doWhile\_r1\_ge\_r2:

mov r3,r3,LSL #1

mov r2,r2,LSL #1

cmp r1,r2

bge doWhile\_r1\_ge\_r2

mov r3,r3,ASR #1

mov r2,r2,ASR #1

pop {lr}

bx lr

.global divmod

divMod:

push {lr}

mov r0,#0

mov r3,#1

cmp r1,r2

blt end

bl scaleleft

bl addsub

end:

pop {lr}

bx lr

.global random

random:

push {lr}

mov r0,#0

bl time

bl srand

bl rand

mov r1,r0,asr #1

/\* bl abs

mov r1,r0\*/

mov r2,#38

bl divMod /\* Call divMod function to get remainder \*/

add r1,#1 /\* Remainder in r1 so add 10 giving between 10 and 99 -> 2 digits \*/

pop {lr}

bx lr

.global printf

.global time

.global srand

.global rand

Gamesinglernum()

.data

in1: .asciz"Which number 1-36 do you bet?\n"

in2:.asciz"How much do you bet?\n"

in3:.asciz"You bet on %d, and you win, you banlance is %d\n\n"

in4:.asciz"You bet on %d, and you loose, you banlance is %d now\n\n"

in5:.asciz"The ball is stay on %d\n"

format: .asciz"%d"

.balign 4

store1: .word 0

.balign 4

store2:.word 0

.text

.global gamesinglenum

gamesinglenum:

push {lr}

ldr r0,addr\_in1

bl printf

ldr r0,addr\_format

ldr r1,addr\_store1

bl scanf @bet number on store1

ldr r0, addr\_in2

bl printf

ldr r0,addr\_format

ldr r1,addr\_store2

bl scanf @bet money on store2

cmp r9,#37

moveq r9,#0

cmp r9,#38

moveq r9,#0

cmp r11,#0

bne counter

ldr r0,addr\_in5

mov r1,r9

bl printf

counter:

ldr r5,addr\_store1

ldr r5,[r5]

ldr r2,addr\_store2

ldr r2,[r2]

mov r3,#35

mul r2,r3,r2

cmp r5,r9

bne run1

add r10,r10,r2

ldr r0,addr\_in3

mov r1,r5

mov r2,r10

bl printf

b end

run1:

ldr r2,addr\_store2

ldr r2,[r2]

sub r10,r10,r2

ldr r0,addr\_in4

mov r1,r5

mov r2,r10

bl printf

end:

pop {lr}

bx lr

addr\_in1:.word in1

addr\_in2:.word in2

addr\_in3:.word in3

addr\_in4:.word in4

addr\_in5:.word in5

addr\_store2:.word store2

addr\_store1:.word store1

addr\_format:.word format

**Gamecolumn()**

.data

in1: .asciz"which Column do you bet?1-3\n"

in2:.asciz"How much do you bet?\n"

in3:.asciz"You bet on Column %d, and you loose ,you banlance is %d now\n\n"

in4:.asciz"You bet on Column %d, and you win, you banlance is %d now\n\n"

in5:.asciz"The ball stay on %d \n"

format:.asciz"%d"

.balign 4

store1:.word 0

.balign 4

store2:.word 0

.text

.global gamecolumn

gamecolumn:

push {lr}

ldr r0,addr\_in1

bl printf

ldr r0, addr\_format

ldr r1,addr\_store1 @number column

bl scanf

ldr r0,addr\_in2

bl printf

ldr r0,addr\_format

ldr r1,addr\_store2 @money

bl scanf

cmp r9,#37

moveq r9,#0

cmp r9,#38

moveq r9,#0

cmp r11,#0

bne counter

ldr r0,addr\_in5

mov r1,r9

bl printf

counter:

mov r5,#0

mov r6,#0

mov r7,#0

mov r8,#0

cmp r9,#1

beq col1

cmp r9,#4

beq col1

cmp r9,#7

beq col1

cmp r9,#10

beq col1

cmp r9,#13

beq col1

cmp r9,#16

beq col1

cmp r9,#19

beq col1

cmp r9,#22

beq col1

cmp r9,#25

beq col1

cmp r9,#28

beq col1

cmp r9,#31

beq col1

cmp r9,#34

beq col1

bne part2 @ball is not in column 1

col1:

mov r5,#1 @ sign the ball stop on column 1

b jump @begin cmpare to the bet

part2:

cmp r9,#2

beq col2

cmp r9,#5

beq col2

cmp r9,#8

beq col2

cmp r9,#11

beq col2

cmp r9,#14

beq col2

cmp r9,#17

beq col2

cmp r9,#20

beq col2

cmp r9,#23

beq col2

cmp r9,#26

beq col2

cmp r9,#29

beq col2

cmp r9,#32

beq col2

cmp r9,#35

beq col2

bne part3

col2:

mov r6,#1 @sign the ball stop on column 2

b jump

part3:

cmp r9,#3

beq col3

cmp r9,#6

beq col3

cmp r9,#9

beq col3

cmp r9,#12

beq col3

cmp r9,#15

beq col3

cmp r9, #18

beq col3

cmp r9,#21

beq col3

cmp r9,#24

beq col3

cmp r9,#27

beq col3

cmp r9,#30

beq col3

cmp r9,#33

beq col3

cmp r9,#36

beq col3

bne part4

col3:

mov r7,#1

b jump

part4:

cmp r9,#0

beq col4

bne jump

col4:

mov r8,#1

b jump

jump:

ldr r4,addr\_store1 @r4 is the column number

ldr r4,[r4]

ldr r3,addr\_store2

ldr r3,[r3]

cmp r8,#1

bne run1

sub r10,r10,r3

ldr r0, addr\_in3

mov r1,r3

mov r2,r10

bl printf

b end

run1:

cmp r4,#1

bne run2

cmp r5,#1

bne out1

add r10,r10,r3

ldr r0,addr\_in4

mov r1,r4

mov r2,r10

bl printf

b end

out1:

sub r10,r10,r3

ldr r0,addr\_in3

mov r1,r4

mov r2,r10

bl printf

b end

run2:

cmp r4,#2

bne run3

cmp r6,#1

bne out1

mov r2,#2

mul r3,r2,r3 @rate\*money

add r10,r10,r3

ldr r0,addr\_in4

mov r1,r4

mov r2,r10

bl printf

b end

run3:

cmp r4,#3

bne run4

cmp r7,#1

bne run4

mov r2,#2

mul r3,r2,r3 @rate\*money

add r10,r10,r3

ldr r0,addr\_in4

mov r1,r4

mov r2,r10

bl printf

b end

run4: @not satisfy 1-3 condition could complete code

b end

end:

pop {lr}

bx lr

addr\_in1:.word in1

addr\_in2:.word in2

addr\_in3:.word in3

addr\_in4:.word in4

addr\_format:.word format

addr\_store1:.word store1

addr\_store2:.word store2

addr\_in5:.word in5

**Gamesingler()**

.data

in1:.asciz"which row do you bet?1-12\n"

in2:.asciz"How much do you bet?\n"

in3:.asciz"You bet on Row %d, and you loose, your banlance is %d now\n\n"

in4:.asciz"You bet on Row %d, and you win, your balance is %now\n\n"

in5:.asciz"The ball stay on %d\n"

.balign 4

store1:.word 0

.balign 4

store2:.word 0

format:.asciz"%d"

.text

.global gamesingler

gamesingler:

push {lr}

ldr r0,addr\_in1

bl printf

ldr r0,addr\_format

ldr r1,addr\_store1

bl scanf @row number

ldr r0, addr\_in2

bl printf

ldr r0,addr\_format

ldr r1,addr\_store2

bl scanf @money

cmp r11,#0

bne counter

ldr r0,addr\_in5

mov r1,r9

bl printf

counter:

ldr r4,addr\_store1

ldr r4,[r4]

ldr r3,addr\_store2

ldr r3,[r3]

cmp r4,#1 @cmp with number of row1

bne run2

cmp r9,#1

blt out1

cmp r9,#3

bgt out1

ble out2

run2:

cmp r4,#2

bne run3

cmp r9,#4

blt out1

cmp r9,#6

bgt out1

ble out2

run3: @cmp with numbers of row3

cmp r4,#3

bne run4

cmp r9,#7

blt out1

cmp r9,#9

bgt out1

ble out2

run4:

cmp r4,#4

bne run5

cmp r9,#10

blt out1

cmp r9,#12

bgt out1

ble out2

run5:

cmp r4,#5

bne run6

cmp r9,#13

blt out1

cmp r9,#15

bgt out1

ble out2

run6:

cmp r4,#6

bne run7

cmp r9,#16

blt out1

cmp r9,#18

bgt out1

ble out2

run7:

cmp r4,#7

bne run8

cmp r9,#19

blt out1

cmp r9,#21

bgt out1

ble out2

run8:

cmp r4,#8

bne run9

cmp r9,#22

blt out1

cmp r9,#24

bgt out1

ble out2

run9:

cmp r4,#9

bne run10

cmp r9,#25

blt out1

cmp r9,#27

bgt out1

ble out2

run10:

cmp r4,#10

bne run11

cmp r9,#28

blt out1

cmp r9,#30

bgt out1

ble out2

run11:

cmp r4,#11

bne run12

cmp r9,#31

blt out1

cmp r9,#33

bgt out1

ble out2

run12:

cmp r4,#12

bne run13

cmp r9,#34

blt out1

cmp r9,#36

bgt out1

ble out2

run13:

b end @when enter is not include 1-12,this things happen,

out1: @bet wrong number

sub r10,r10,r3

ldr r0,addr\_in3

mov r1,r4

mov r2,r10

bl printf

b end

out2:

mov r2,#5

mul r3,r2,r3 @bet correct number

add r10,r10,r3

ldr r0,addr\_in4

mov r1,r4

mov r2,r10

bl printf

b end

end:

pop {lr}

bx lr

addr\_in1:.word in1

addr\_in2:.word in2

addr\_in3:.word in3

addr\_in4:.word in4

addr\_in5:.word in5

addr\_format:.word format

addr\_store1:.word store1

addr\_store2:.word store2

**Gamecolor()**

.data

in1:.asciz"What color do you bet on r/b ?\n"

in2:.asciz"How much do you bet? \n"

in3:.asciz"You bet on Color %s, and you loose,your banlance is %d now\n\n"

in4:.asciz"You bet on Color %s, and you win, your balance is %d now\n\n"

in5:.asciz"The ball stay on %d\n"

format1:.asciz"%d"

format2:.asciz"%s"

cmp1:.asciz"r"

cmp2:.asciz"b"

.balign 4

store1:.word 0

.balign 4

store2:.word 0

.text

.globl gamecolor

gamecolor:

push {lr}

ldr r0,addr\_in1

bl printf

ldr r0,addr\_format2

ldr r1,addr\_store1

bl scanf @color b/r

ldr r0,addr\_in2

bl printf

ldr r0, addr\_format1

ldr r1,addr\_store2

bl scanf @money

cmp r11,#0

bne counter

ldr r0,addr\_in5

mov r1,r9

bl printf

counter:

ldr r0,addr\_cmp1 @assume r

ldr r1,addr\_store1

bl strcmp

bne jump

cmp r9,#1

bne run2

beq out2

run2:

cmp r9,#3

bne run3

beq out2

run3:

cmp r9,#5

bne run4

beq out2

run4:

cmp r9,#7

bne run5

beq out2

run5:

cmp r9,#9

bne run6

beq out2

run6:

cmp r9,#12

bne run7

beq out2

run7:

cmp r9,#14

bne run8

beq out2

run8:

cmp r9,#16

bne run9

beq out2

run9:

cmp r9,#18

bne run10

beq out2

run10:

cmp r9,#19

bne run11

beq out2

run11:

cmp r9,#21

bne run12

beq out2

run12:

cmp r9,#23

bne run13

beq out2

run13:

cmp r9,#25

bne run14

beq out2

run14:

cmp r9,#27

bne run15

beq out2

run15:

cmp r9,#30

bne run16

beq out2

run16:

cmp r9,#32

bne run17

beq out2

run17:

cmp r9,#34

bne run18

beq out2

run18:

cmp r9,#36

bne out1

beq out2

jump:

ldr r0,addr\_cmp2

ldr r1,addr\_store1

bl strcmp

bne jump2

cmp r9,#2

bne run\_2

beq out2

run\_2:

cmp r9,#4

bne run\_3

beq out2

run\_3:

cmp r9,#6

bne run\_4

beq out2

run\_4:

cmp r9,#8

bne run\_5

beq out2

run\_5:

cmp r9,#10

bne run\_6

beq out2

run\_6:

cmp r9,#11

bne run\_7

beq out2

run\_7:

cmp r9,#13

bne run\_8

beq out2

run\_8:

cmp r9,#15

bne run\_9

beq out2

run\_9:

cmp r9,#17

bne run\_10

beq out2

run\_10:

cmp r9,#20

bne run\_11

beq out2

run\_11:

cmp r9,#22

bne run\_12

beq out2

run\_12:

cmp r9,#24

bne run\_13

beq out2

run\_13:

cmp r9,#26

bne run\_14

beq out2

run\_14:

cmp r9,#28

bne run\_15

beq out2

run\_15:

cmp r9,#29

bne run\_16

beq out2

run\_16:

cmp r9,#31

bne run\_17

beq out2

run\_17:

cmp r9,#33

bne run\_18

beq out2

run\_18:

cmp r9,#35

bne out1

beq out2

jump2:

b end @other possibility ex: wrong input.

out1:

ldr r3,addr\_store2

ldr r3,[r3]

sub r10,r10,r3 @loose

ldr r0,addr\_in3

ldr r1,addr\_store1

mov r2,r10

bl printf

b end

out2:

ldr r3,addr\_store2

ldr r3,[r3] @win

add r10,r10,r3

ldr r0,addr\_in4

ldr r1,addr\_store1

mov r2,r10

bl printf

b end

end:

pop {lr}

bx lr

addr\_in1:.word in1

addr\_in2:.word in2

addr\_in3:.word in3

addr\_in4:.word in4

addr\_in5:.word in5

addr\_format1:.word format1

addr\_format2:.word format2

addr\_cmp1:.word cmp1

addr\_cmp2:.word cmp2

addr\_store1:.word store1

addr\_store2:.word store2

.global strcmp

**Gameoe()**

.data

in1:.asciz"Do you bet odd/even? input o/e\n"

in2:.asciz"How much do you bet? \n"

in3:.asciz"You bet on %s (odd/even), and you loose,your banlance is %d now\n\n"

in4:.asciz"You bet on %s (odd/even), and you win, your balance is %d now\n\n"

in5:.asciz"The ball is on %d\n"

format1:.asciz"%d"

format2:.asciz"%s"

cmp1:.asciz"o"

cmp2:.asciz"e"

.balign 4

store1:.word 0

.balign 4

store2:.word 0

.text

.globl gameoe

gameoe:

push {lr}

ldr r0,addr\_in1

bl printf

ldr r0,addr\_format2

ldr r1,addr\_store1

bl scanf @odd or even o/e

ldr r0,addr\_in2

bl printf

ldr r0, addr\_format1

ldr r1,addr\_store2

bl scanf @money

cmp r11,#0

bne counter

ldr r0,addr\_in5

mov r1,r9

bl printf

counter:

cmp r9,#37

bne jump1

beq out1

jump1:

cmp r9,#38

bne jump2

beq out1

jump2:

mov r5,#3 @judgement signal

ldr r0,addr\_cmp1

ldr r1,addr\_store1

bl strcmp

bne run1 @is not odd

mov r5,#1

b compare

run1:

ldr r0,addr\_cmp2

ldr r1,addr\_store1

bl strcmp

bne run2

mov r5,#2

b compare

run2:

b end @input is not o,or e

compare:

ldr r3,addr\_store2

ldr r3,[r3]

mov r6,#3

and r9,r9,#1

cmp r9,#0

moveq r6,#2

movne r6,#1

cmp r5,r6

beq out2

bne out1

out1:

sub r10,r10,r3 @loose

ldr r0,addr\_in3

ldr r1,addr\_store1

mov r2,r10

bl printf

b end

out2: @win

add r10,r10,r3

ldr r0,addr\_in4

ldr r1,addr\_store1

mov r2,r10

bl printf

b end

end:

pop {lr}

bx lr

addr\_in1:.word in1

addr\_in2:.word in2

addr\_in3:.word in3

addr\_in4:.word in4

addr\_in5:.word in5

addr\_format1:.word format1

addr\_format2:.word format2

addr\_cmp1:.word cmp1

addr\_cmp2:.word cmp2

addr\_store1:.word store1

addr\_store2:.word store2