**Project 2**  
**<Crown and Anchor>**

**CSC-11-48982**  
 **Name: Najera, Enrique**  
 **Date: 16 November 2015**

**Introduction**

Title: Crown and Anchor

Crown and Anchor is a British dice game mainly played in pubs.

The gambler selects an integer from 1 to 6 and the player places a bet. Then three dice are rolled from the *Bird Cage*. If exactly x dice show the gambler’s integer, the payoff is x to 1; else, the player loses their bet.

I am doing this style of project because it easily uses all the concepts we have used in class thus far.

The importance of this is to expand on what I already knew and use what I know in new ways.

**Summary**

Project size: ~573 lines assembly, ~167 lines C

Variables: 5

Concepts: ~22

I would say it was a bit troublesome, but not as much as the first project because I had more experience. I had the most trouble trying to overwrite variables (like project 1) which I tried by using scanf tricks but they ended up getting overwritten way before I needed them. I also had a lot of trouble trying to use floats with non-floats (which was a must for this project to actually do anything). Problem being I stored almost every value in non-float registers. I also had one bug where, after placing a bet, the program become slows down and appears as it is unresponsive, or crashed. I figured it was a loop somewhere too deep in the code to catch.

**Description**

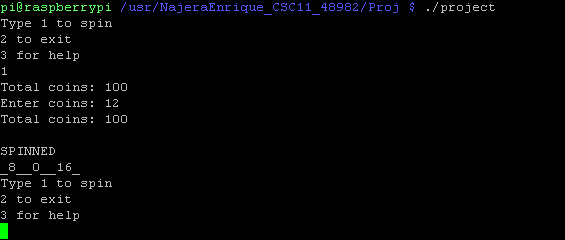
I first wrote the C code as close to assembly as possible, which helped a lot on the layout and shifting (which eventually didn’t get used much in the end). After, I made a menu that goes to other files and made sure it worked perfectly. Then I worked heavily on gSelect.s which ended up being the main game’s “logic holder.” Lastly, I tried to implement floats and float interaction which didn’t work as I hoped.

**Sample Input/Output**

Input 1 to spin. Prompts for coins.



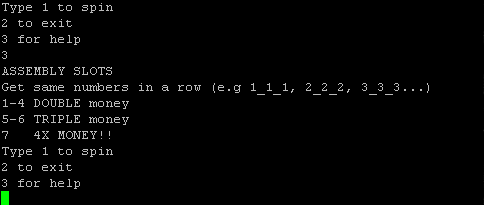
After coin input, starts a spin.



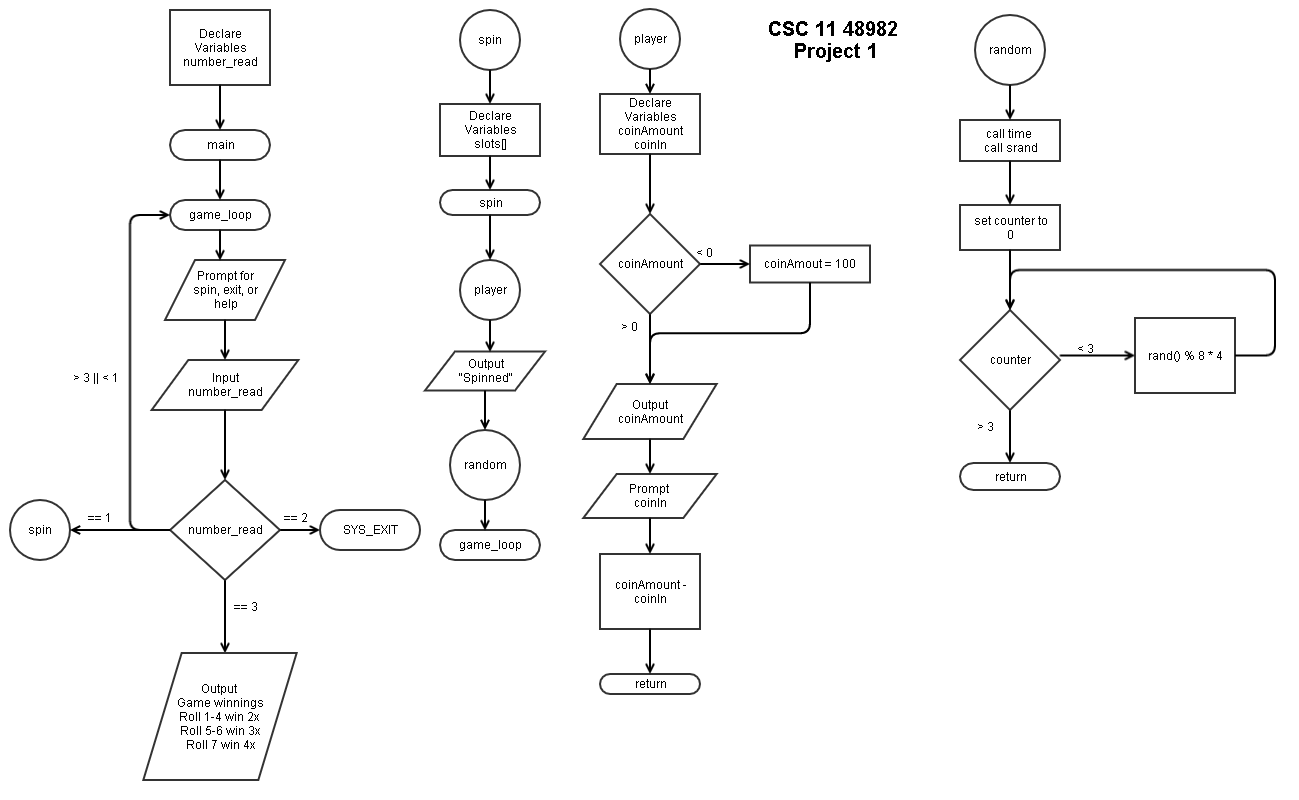
Input 2 to exit program.



Input 3 to see help menu (what you can win).



**Flowchart**



**Pseudo-code**

*While player choice < 3*

*Prompt for roll, help, or exit*

*Display player’s wallet*

*If player’s wallet <= 0.00*

*Display out of money error*

*If prompt = 1*

*Prompt for bet*

*If bet more than wallet amount*

*Display bet more than wallet error*

*Else*

*Gambler selects integer random number 1-6*

*Spin dice random number 1-6*

*If die matches gambler integer*

*Increment counter*

*Output win amount increment counter*

*If counter returns 0*

*Wallet -= bet*

*Else*

*Multiply bets by increment counter*

*Add winnings to wallet*

*If player rolled 3 1 4*

*Output surprise message*

*Add 3.14 to wallet*

*If prompt = 2*

*Display help page*

*Prints About and How to Play*

*If prompt >= 3*

*Terminate program successfully*

**Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **VARIABLE** | **PURPOSE** | **FILE** | **LINE NUMBER(S)** |
| bet | Holds user’s bet | gSelect.s | 22, 48, 52-53, 55 |
| gArray | Holds int selection numbers | gSelect.s | 23 |
| selection | Holds gambler’s int selection | gSelect.s | 24 |
| choice | Holds user’s menu choice | main.s | 18, 39, 40, 43-44, 47, 52 |
| walletAmt | Holds user’s current money | wallet.s | 13, 21-23, 25-27 |

**Language Constructs**

|  |  |  |
| --- | --- | --- |
| **CONSTRUCT** | **FILE** | **LINE NUMBER(S)** |
| string | errorMsgs.s | 13, 14 |
|  | gSelect.s | 12-16 |
|  | help.s | 14-22, 26-29 |
|  | main.s | 12-14 |
|  | randomDie.s | 12 |
|  | random.s | 12 |
|  | wallet.s | 13 |
| Global variables | gSelect.s | 22, 24, 52-53, 54 |
|  | main.s | 18, 19, 43-44 |
|  | wallet.s | 17 |
| Floats | wallet.s | 17, 22-23, 26 |
| printf | errorMsgs.s | 23-24, 29-30 |
|  | help.s | 36-53, 56-63 |
|  | main.s | 35 |
|  | wallet.s | 25-27 |
| Printing floats | wallet.s | 25-27 |
| scanf | gSelect.s | 47-49 |
|  | main.s | 38-40 |
| time | randomDie.s | 22 |
|  | random.s | 22 |
| srand | randomDie.s | 23 |
|  | random.s | 23 |
| Global labeling | errorMsgs.s | 18-20 |
|  | gSelect.s | 28-30 |
|  | main.s | 23-25 |
|  | randomDie.s | 16 |
|  | random.s | 16 |
|  | wallet.s | 19 |
| Labels | main.s | 29, 32, 58 |
|  | errorMsgs.s | 21, 22, 28 |
|  | gSelect.s | 31, 36, 66, 72, 85 |
|  | help.s | 34 |
|  | randomDie.s | 17, 29, 48, 58, 62, 74, 86 |
|  | random.s | 17, 29, 48, 58, 62, 74, 86 |
|  | wallet.s | 20 |
| loops | randomDie.s | 29, 64, 76, 89 |
|  | random.s | 29, 64, 76, 89 |
| Stack (PUSH, POP) | gSelect.s | 32 |
|  | randomDie.s | 45, 49, 59, 63, 71, 75, 83, 87, 95 |
|  | random.s | 45, 49, 59, 63, 71, 75, 83, 87, 95 |
|  | main.s | 27, 60 |
| External labels | errorMsgs.s | 26, 30, 38 |
|  | gSelect.s | 120-123 |
|  | help.s | 87, 88 |
|  | main.s | 72-73 |
|  | randomDie.s | 102-107 |
|  | random.s | l02-106 |
|  | wallet.s | 35, 36 |
| CMP | gSelect.s | 55, 68, 74, 76, 78, 81, 97 |
|  | main.s | 47, 52 |
|  | randomDie.s | 41, 53, 67, 80, 92 |
|  | random.s | 41, 53, 67, 80, 92 |
| Branching | errorMsgs.s | 26, 31 |
|  | gSelect.s | 34, 56, 63, 69, 82, 98, 106 |
|  | help.s | 65 |
|  | main.s | 31, 48, 49, 53, 54, 56 |
|  | randomDie.s | 30, 34, 42, 44, 54-56, 68, 79, 81, 93 |
|  | random.s | 30, 34, 42, 44, 54-56, 68, 79, 81, 93 |
|  | wallet.s | 29 |
| ADDEQ | gSelect.s | 75, 77, 79 |
| Logical Shift | gSelect.s | 103, 65, 66 |
|  | randomDie.s | 65, 66 |
|  | random.s | 65, 66 |
| Arithmetic shift | randomDie.s | 31, 69, 70 |
|  | random.s | 31, 69, 70 |
| Referencing | errorMsgs.s | 34, 35 |
|  | gSelect.s | 109-117 |
|  | main.s | 64-66, 68, 69 |
|  | help.s | 69-77, 81-84 |
|  | randomDie.s | 99 |
|  | random.s | 99 |
|  | wallet.s | 32 |
| Scaled int | gSelect.s | 102 |
| arrays | gSelect.s | 23 |
| “%f” | wallet.s | 13 |
| VMOV | gSelect.s | 93 |
|  | wallet.s | 26 |
| VSUB | gSelect.s | 94 |
| VADD | gSelect.s | 104 |
| VLDR | wallet.s | 22 |
| VCVT | wallet.s | 23 |

**References**

Got random function from your upload to github. Everything else I copied and modified from previous assignments I have completed. Also looked into thinkinggeek website for help and structure.

**Program**

**main.c**

/\*

\* File: main.c

\* Author: Enrique Najera

\* Purpose: Crown and Anchor in c

\* 16 December 2015

\*/

// SYS\_LIBS

#include <stdio.h>

#include <time.h> // time()

#include <stdlib.h> // rand()

// Function Prototypes

int gSelect();

float birdCage(int, float, float);

void help();

// Start method main

int main(void)

{

// Declare Variables

int R1 = 0, // Gambler's int selection

R2 = 0x3243F6A9, // $3.14

R3 = 0; // Player's choice

// -1 Roll 2 about 3+ exit

float R4 = 50.0f, // Start player at $50

R5 = 0.0f; // Holds player's bet

// Display Menu

do

{

// Display choices and prompt for input

printf("\_\_\_\_CROWN AND ANCHOR\_\_\_\_\n"

"1) Roll\n2) Help\n3) Exit\n"

"Wallet: %.2f\n", R4);

scanf("%d", &R3);

// If out of money, exit

if (R4 <= 0.0f)

{

printf("\nOut of money!!!\n");

R3 = 3;

}

// Roll

if (R3 == 1)

{

// Prompt for bet

printf("Place your bet: ");

scanf("%f", &R5);

// Check if player can bet

if (R5 > R4)

{

printf("Bet more than wallet!!\n");

}

else

{

R1 = gSelect(); // Display Gambler's Selection

R4 = birdCage(R1, R4, R5); // Spin the Bird Cage

}

}

// Help

else if (R3 == 2)

help();

// Exit

}while(R3 < 3);

printf("\nGOODBYE!\n");

// SYS\_EXIT

return 0;

}// End method main

// Start method gSelect

int gSelect()

{

// Declare Variables

int R0; // Holds random number

// Set random number seed

srand(time(NULL));

// Rand num from 1 to 6

R0 = rand() % 6 + 1;

return R0;

}// End method gSelect

// Start method birdCage

float birdCage(int R0, float R4, float R5)

{

// Declare Variables

int R1 = 0, // Die 1

R2 = 0, // Die 2

R3 = 0, // Die 3

R6 = 0; // Counter for rolls

printf("\n Gambler says %d\n", R0);

// Set random number seed

//srand(time(NULL));

R1 = rand() % 6 + 1;

R2 = rand() % 6 + 1;

R3 = rand() & 6 + 1;

// Output spins

printf("\nSPINNED %d %d %d\n", R1, R2, R3);

// Check rolls

if (R1 == R0)

R6 = R6 + 1;

if (R2 == R0)

R6 = R6 + 1;

if (R3 == R0)

R6 = R6 + 1;

// Output win amount

printf("Bet multiplied by: %d\n", R6);

// If rolls dont match, wallet -= bet

if (R6 == 0)

{

R4 = R4 - R5;

}

else

{

// Do math

R5 = R6 \* R5; // Multiply bet by rolls

R4 = R5 + R4; // Add bet to total

}

// If player got 3 1 4

if (R1 == 3 && R2 == 1 && R3 == 4)

{

printf("\n GOT EXTRA $3.14!!!\n");

R0 = 0x3243F6A9; // pi

R0>>=28;

R4 = R0 + R4;

}

// Return total amount

return R4;

}// End method birdCage

// Start method help

void help()

{

// Print about

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_ABOUT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

"--------CROWN AND ANCHOR--------\n"

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

"The gambler selects an integer\n"

"from 1 to 6, and then three dice\n"

"are rolled. If exactly x dice show\n"

"the gambler's number, the payoff is\n"

"x to 1.\nElse, lose your bet\n"

"SPIN 3 1 4 FOR A FREE $3.14!!\n"

);

// Print how to play

printf("\n\_\_\_\_\_\_\_\_HOW TO PLAY\_\_\_\_\_\_\_\_\_\_\_\n"

"After choosing to roll, the gambler\n"

"will say their integer then you\n"

"place your bet and hit Return/Enter.\n"

);

}// End method help

**main.s**

/\*

File: main.s

Author: Enrique Najera

Purpose: CSC 11 Project 2

Main menu

\*/

.data

/\* Messages \*/

.balign 4

msgPrompt: .asciz "\_\_\_\_CROWN AND ANCHOR\_\_\_\_\n1) Roll\n2) Help\n3) Exit\n"

msgNoMoney: .asciz "\nOut of money!!!\n"

msgExit: .asciz "\nGOODBYE!\n"

/\* Variables \*/

.balign 4

choice: .word 0

format: .asciz "%d"

.text

.global main

.global menu

.global menuOut

main:

PUSH {R4, LR}

menu:

// Display wallet

B wallet

menuOut:

// Display choices

LDR R0, address\_of\_msgPrompt

BL printf

// Get choice

LDR R0, address\_of\_format

LDR R1, address\_of\_choice

BL scanf

// Store number for comparing

LDR R1, address\_of\_choice

LDR R1, [R1]

// Roll

CMP R1, #1

BEQ gSelect

BLT exit

// Help/About menu

CMP R1, #2

BEQ help

BGT exit

BAL menu

exit:

// SYS\_EXIT

POP {R4, LR}

BX LR

/\* References \*/

address\_of\_msgPrompt: .word msgPrompt

address\_of\_msgNoMoney: .word msgNoMoney

address\_of\_msgExit: .word msgExit

address\_of\_choice: .word choice

address\_of\_format: .word format

/\* Extern \*/

.global printf

.global scanf

**random.s**

/\*

File: random.s

Author: Enrique Najera

Purpose: CSC 11 Project 2

Generates random number

18 December 2015

\*/

.data

.balign 4

debugMsg: .asciz "RAND: %d\n"

.text

.global random

random:

/\* Generate random number \*/

PUSH {R4, LR}

MOV R0, #0

BL time

BL srand

MOV R4, #0

MOV R5, #1 @Get 1-6

/\* rand() % 6 + 1 \*/

loop\_rand:

BL rand

MOV R1, R0, ASR #1

MOV R2, #6

BL divMod

ADD R1, R5

/\* LDR R0, address\_of\_debugMsg

BL printf\*/

ADD R4, #1

CMP R4, #1 @only get one random number

BLT loop\_rand

B gOut

POP {R4, LR}

BX LR

divMod:

PUSH {LR}

MOV R0, #0

MOV R3, #1

CMP R1, R2

BLT end

BL scaleLeft

BL addSub

end:

POP {LR}

BX LR

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

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

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

/\* References \*/

address\_of\_debugMsg: .word debugMsg

/\* Extern \*/

.global printf

.global time

.global srand

.global rand

.global menu

**randomDie.s**

/\*

File: randoem

Author: Enrique Najera

Purpose: CSC 11 Project 2

Generates random number for dice

18 December 2015

\*/

.data

.balign 4

debugMsg: .asciz "RAND: %d\n"

.text

.global randomDie

randomDie:

/\* Generate random number \*/

//PUSH {R4, LR}

MOV R0, #0

BL time

BL srand

MOV R4, #0

MOV R5, #1 @Get 1-6

/\* rand() % 6 + 1 \*/

loop\_rand:

BL rand

MOV R1, R0, ASR #1

MOV R2, #6

BL divMod

ADD R1, R5

/\* LDR R0, address\_of\_debugMsg

BL printf\*/

ADD R4, #1

CMP R4, #1 @only get one random number

BLT loop\_rand

B dieCheck

POP {R4, LR}

BX LR

divMod:

PUSH {LR}

MOV R0, #0

MOV R3, #1

CMP R1, R2

BLT end

BL scaleLeft

BL addSub

end:

POP {LR}

BX LR

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

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

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

/\* References \*/

address\_of\_debugMsg: .word debugMsg

/\* Extern \*/

.global printf

.global time

.global srand

.global rand

.global menu

.global dieCheck

**wallet.s**

/\*

File: wallet.s

Author: Enrique Najera

Purpose: CSC 11 Project 2

Holds wallet amount as float

16 December 2015

\*/

.data

/\* Messages \*/

.balign 4

walletMsg: .asciz "Wallet: %.2f\n"

/\* Declare Variables \*/

.balign 4

walletAmt: .float 50.0

.global wallet

wallet:

LDR R1, address\_of\_walletAmt

VLDR S14, [R1]

VCVT.F64.F32 D5, S14

LDR R0, =walletMsg

VMOV R2, R3, D5

BL printf

B menuOut

/\* References \*/

address\_of\_walletAmt: .word walletAmt

/\* Extern \*/

.global printf

.global menuOut

**errorMsgs.s**

/\*

File: errorMsgs.s

Author: Enrique Najera

Purpose: CSC 11 Project 2

Holds all error messages

16 December 2015

\*/

.data

/\* Messages \*/

.balign 4

error\_betExceedsMsg: .asciz "Bet more than wallet!!\n"

error\_outOfMoneyMsg: .asciz "\nOut of money!!!\n"

.text

.global errorMsgs

.global error\_betExceeds

.global error\_outOfMoney

errorMsgs:

error\_betExceeds:

LDR R0, address\_of\_error\_betExceedsMsg

BL printf

B menu

error\_outOfMoney:

LDR R0, address\_of\_error\_outOfMoneyMsg

BL printf

B menu

/\* References \*/

address\_of\_error\_betExceedsMsg: .word error\_betExceedsMsg

address\_of\_error\_outOfMoneyMsg: .word error\_outOfMoneyMsg

/\* Extern \*/

.global menu

**gSelect.s**

/\*

File: gSelect.s

Author: Enrique Najera

Purpose: CSC 11 Project 2

Creates gambler's selection

\*/

.data

/\* Messages \*/

.balign 4

gamblerMsg: .asciz "\n Gambler says %d\n"

betMsg: .asciz "Place your bet: "

rollMsg: .asciz "\nSPINNED %d\n"

multMsg: .asciz "Bet multiplied by: %d\n"

piMsg: .asciz "\n GOT EXTRA $3.14!!!\n"

scan: .asciz "%d"

/\* Declare Variables \*/

.balign 4

bet: .word 0

gArry: .word 1,2,3,4,5,6

selection: .word 0

.text

.global gSelect

.global gOut

.global dieCheck

gSelect:

PUSH {R4, LR}

B random

gOut:

// Hold selection in static reg

MOV R6, R1

// Print selection

LDR R0, address\_of\_gamblerMsg

BL printf

// Prompt for bet

LDR R0, address\_of\_betMsg

BL printf

LDR R0, address\_of\_scan

LDR R1, address\_of\_bet

BL scanf

/\* Store bet in R0 for checking \*/

LDR R2, address\_of\_bet

LDR R2, [R2]

CMP R2, #50

BGT error\_betExceeds

/\* INIT vars \*/

MOV R2, #0 @Counter for multiplying

MOV R3, #0 @counter for 3.14

MOV R4, #0 @Loop counter

B die

// INIT dice

die:

ADD R4, R4, #1

CMP R4, #4

BEQ mult

B randomDie

dieCheck:

/\* Check for 3 1 4 \*/

CMP R1, #3

ADDEQ R3, R3, #1

CMP R1, #1

ADDEQ R3, R3, #1

CMP R1, #4

ADDEQ R3, R3, #1

/\* Check for gambler's integer \*/

CMP R1, R6

BNE die

ADD R2, R2, #1

mult:

/\* MAKES PROGRAM SLOOOOOW

LDR R0, address\_of\_rollMsg

BL printf

If rolls dont match, wallet -= bet

CMP R3, #0

VMOV R0, R1, D5

VSUB.F64 D5, D0, D5\*/

// If player got 3 1 4

CMP R3, #3

BNE menu

LDR R0, address\_of\_piMsg

BL printf

LDR R0, =0x3243F6A9 @Pi BP-32 WD32

LSR R0, #28 @ BP-04 WD04

//VADD.F64 D5, D5, R0

B menu

/\* References \*/

address\_of\_gamblerMsg: .word gamblerMsg

address\_of\_betMsg: .word betMsg

address\_of\_rollMsg: .word rollMsg

address\_of\_piMsg: .word piMsg

address\_of\_gArry: .word gArry

address\_of\_selection: .word selection

address\_of\_multMsg: .word multMsg

address\_of\_bet: .word bet

address\_of\_scan: .word scan

/\* Extern \*/

.global menu

.global printf

.global randomDie

.global error\_betExceeds

**help.s**

/\*

File: roll.s

Author: Enrique Najera

Purpose: CSC 11 Project 2

Holds roll properties

18 December 2015

\*/

.data

/\* Messages by line for readability\*/

/\*\*ABOUT \*/

.balign 4

aboutMsgL1: .asciz "\_\_\_\_\_\_\_\_\_\_\_\_\_ABOUT\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

aboutMsgL2: .asciz "--------CROWN AND ANCHOR--------\n"

aboutMsgL3: .asciz "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

aboutMsgL4: .asciz "The gambler selects an integer\n"

aboutMsgL5: .asciz "from 1 to 6, and then three dice\n"

aboutMsgL6: .asciz "are rolled. If exactly x dice show\n"

aboutMsgL7: .asciz "the gambler's number, the payoff is\n"

aboutMsgL8: .asciz "x to 1.\nElse, lose your bet\n"

aboutMsgL9: .asciz "SPIN 3 1 4 FOR A FREE $3.14!!\n"

/\*\*HOW TO PLAY \*/

.balign 4

howToMsgL1: .asciz "\n\_\_\_\_\_\_\_\_HOW TO PLAY\_\_\_\_\_\_\_\_\_\_\_\n"

howToMsgL2: .asciz "After choosing to roll, the gambler\n"

howToMsgL3: .asciz "will say their integer then you\n"

howToMsgL4: .asciz "place your bet and hit Return/Enter.\n"

.text

.global help

help:

// Print ABOUT

LDR R0, address\_of\_aboutMsgL1

BL printf

LDR R0, address\_of\_aboutMsgL2

BL printf

LDR R0, address\_of\_aboutMsgL3

BL printf

LDR R0, address\_of\_aboutMsgL4

BL printf

LDR R0, address\_of\_aboutMsgL5

BL printf

LDR R0, address\_of\_aboutMsgL6

BL printf

LDR R0, address\_of\_aboutMsgL7

BL printf

LDR R0, address\_of\_aboutMsgL8

BL printf

LDR R0, address\_of\_aboutMsgL9

BL printf

// Print HOW TO PLAY

LDR R0, address\_of\_howToMsgL1

BL printf

LDR R0, address\_of\_howToMsgL2

BL printf

LDR R0, address\_of\_howToMsgL3

BL printf

LDR R0, address\_of\_howToMsgL4

BL printf

B menu

/\* References \*/

/\*\*ABOUT \*/

address\_of\_aboutMsgL1: .word aboutMsgL1

address\_of\_aboutMsgL2: .word aboutMsgL2

address\_of\_aboutMsgL3: .word aboutMsgL3

address\_of\_aboutMsgL4: .word aboutMsgL4

address\_of\_aboutMsgL5: .word aboutMsgL5

address\_of\_aboutMsgL6: .word aboutMsgL6

address\_of\_aboutMsgL7: .word aboutMsgL7

address\_of\_aboutMsgL8: .word aboutMsgL8

address\_of\_aboutMsgL9: .word aboutMsgL9

/\*\*HOW TO PLAY \*/

address\_of\_howToMsgL1: .word howToMsgL1

address\_of\_howToMsgL2: .word howToMsgL2

address\_of\_howToMsgL3: .word howToMsgL3

address\_of\_howToMsgL4: .word howToMsgL4

/\* Extern \*/

.global printf

.global menu