===

upload\_mode:

;@ zero out Zmem and set upload pointer to 0

LDR R2, =Zmem

LDR R1, [R2]

LDR R8, =0xE0001030

LDR R9, [R8, #0]

STR R9, [R8, #0]

check\_buffer:

LDR R8, =0xE000102C ;@ address for UART1 - XUARTPS\_SR\_OFFSET

LDR R9, [R8, #0]

MOV R10, #8

TST R9, R10 ;@ Check 4th bit to see if Transmitter FIFO is empty

BNE check\_buffer

check\_sliders:

LDR R8, =0x41220000 ;@ address for slider switches

LDR R9, [R8, #0]

LDR R10, #128

TST R9, R10

BEQ upload\_mode

LDR R10, [R8, #0]

CMP R10, R9 ;@ compare new with old

BLEQ delay\_loop

BEQ check\_sliders

BNE which\_slider

which\_slider:

LDR R8, =0x41220000 ;@ address for slider switches

LDR R9, [R8, #0]

LDR R11, [R3, #0] ;@ previous slider value

delay\_loop:

LDR R10, =500000

loop:

SUBS R10, R10, #1

BNE loop

MOV R15, R14

===

.global asm\_main

.align 4

memory: .space 2000000 ;@ 2MB reserved for own use throughout program

Zmem: .space 2000000 ;@ 2MB for uploading Zprogram

Zstack: .space 1000000 ;@ 1MB for Zstack

FZreg: .space 1000000 ;@ 1MB for function local Zregisters

temp: .space 256 ;@ 256B temporary storage for holding temp operands to variable operand count instructions

.align 4

asm\_main:

;@ Set up transmitter to what TeraTerm is expecting

;@ receiver is already set up

LDR R9, =0x20

LDR R4, =0xE0001004 ;@ Mode Register -- Tera Term transmitter

STR R9, [R4, #0]

;@ Start Baud rate setup

LDR R9, =0x3E ;@ 62 in decimal value

LDR R4, =0xE0001018 ;@ Baud rate generator

STR R9, [R4, #0]

;@ Complete baud rate setup

LDR R9, =0x6

LDR R4, =0xE0001034 ;@ Baud rate divider

STR R9, [R4, #0]

;@ Enable and reset the UART

LDR R9, =0x117

LDR R4, =0xE0001000 ;@ Control register

STR R9, [R4, #0]

check\_switch:

PUSH {R14}

LDR R0, =0x41220000 ;@ load address for slider switches

CMP R0, #128 ;@ SW07 - Up for upload and down for run mode

BEQ upload\_mode

CMP R0, #64 ;@ SW06 - no-header mode (raw Zcode)

BEQ run\_mode

CMP R0, #32 ;@ SW05 - debug

BEQ debug\_mode

POP {R15}

upload\_mode:  
 LDR R10, =Zmem  
 LDR R11, =2000000 ;@ Size of Zmem  
 LDR R8, =0 ;@ Zmem offset incrementer  
 LDR R9, =0 ;@ the null value to store at Zmem offset by R8  
 LDR R12, =1000000 ;@ Size of Zstack  
 BL Delete\_Zmemory  
 BL Reset\_Zstack  
 BL Check\_receiver\_fifo\_and\_store  
 BL Check\_transmitter\_fifo\_and\_print  
 B \_done  
  
 \_done:  
 B \_done  
  
  
Check\_receiver\_fifo\_and\_store:  
 LDR R11, =2 ;@ 2  
 LDR R8, =0xE0001030 ;@ Memory location where hello world is stored  
 LDR R9, =0xE000102C ;@ XUARTPS\_SR\_OFFSET  
 LDR R9, [R9, #0]  
 ;@TST R9, #0b00000010  
 AND R11, R9, R11  
 CMP R11, #2 ;@ Check 1st bit to see if Transmitter FIFO is empty  
 BEQ Store  
 BNE Check\_receiver\_fifo\_and\_store  
  
Store:  
 LDR R9, [R7, #0] ;@ Loading the value of the fifo receiver store counter to R9  
 LDRB R8, [R8, #0] ;@ loading the input from UART memory location to R8  
 ADD R9, R9, #4  
 STRB R8, [R7, R9] ;@ Storing  
 STR R9, [R7, #0] ;@ Update the value of the fifo receiver store counter  
 MOV PC, LR  
  
Check\_transmitter\_fifo\_and\_print:  
 LDR R11, =16384 ;@ 2^14  
 LDR R8, =0xE0001030 ;@ Memory location where hello world is stored  
 LDR R9, =0xE000102C ;@ XUARTPS\_SR\_OFFSET  
 LDR R9, [R9, #0]  
 ;@TST R9, R11  
 AND R11, R9, R11  
 CMP R11, #16384 ;@ Check 14th bit to see if Transmitter FIFO is empty  
 BNE Check\_transmitter\_fifo\_and\_print  
 LDREQ R11, =4  
 BEQ print  
  
print:  
 LDR R9, [R7, #0]  
 LDR R10, [R7, R11]  
 ADD R11, R11, #4  
 CMP R11, R9  
 MOVEQ PC, LR  
 STR R10, [R8, #0]  
  
Delete\_Zmemory:  
 STR R9, [R10, R8]  
 ADD R8, R8, #4  
 CMP R8, R11  
 BNE Delete\_Zmemory  
 LDREQ R8, =0  
 MOVEQ PC, LR  
  
Reset\_Zstack:  
 STR R9, [R10, R8]  
 ADD R8, R8, #4  
 CMP R8, R12  
 BNE Delete\_Zmemory  
 LDREQ R8, =0  
 MOVEQ PC, LR

decode\_instructions\_loop:

LDR R8, [R7, R4] ;@ Fetch byte from Zmem at ZPC -- Opcode

ADD R4, R4, #1

LDR R10, =192 ;@ bits 7 & 6

AND R9, R8 R10

CMP R9, #128 ;@ A-Type - 10

BLEQ a\_type

CMP R9 #192 ;@ C-Type - 11

BLEQ c\_type

BLNE b\_type ;@ 01

MOV R15, LR

a\_type:

LDR R10, =48 ;@ bits 5 & 4

AND R11, R8, R10

STR R11, [=memory, #100]

CMP R11, #48 ;@ operand count is zero - 11

CMP R11, #0 ;@ two byte constant - 00

CMP R11, #16 ;@ one byte constant - 01

CMP R11, #32 ;@ one byte register indicator - 10

LDR R12, =15 ;@ instruction indicator -- bits 3 - 0

AND R9, R8, R12

STR R9, [=memory, #200]

b\_type:

LDR R10, =64

AND R11, R8, R10

STR R11, [=memory, #300]

CMP R11, #0 ;@ 1st operand: one byte constant

CMP R11, #64 ;@ 1st operand: operand in a register

LDR R10, =32

AND R11, R8, R10

STR R11, [=memory, #400]

CMP R11, #0 ;@ 2nd operand: one byte constant

CMP R11, #32 ;@ 2nd operand: operand in register

c\_type:

LDR R10, =32

AND R11, R8, R10

STR R11, [=memory, #500]

CMP R11, #0 ;@ two operands

CMP R11, #32 ;@ variable operand count

LDR R10, =31

AND R11, R8, R10

STR R11, [=memory, #600] ;@ instruction indicator

Decode\_text:

Store\_Opcodes:

LDRB R0, =Zmem

MOV R2, #0

LDR R9, =0x0 ;@ RTRUE

BL inc\_counter

LDR R9, =0x1 ;@ RFALSE

BL inc\_counter

LDR R9, =0x2 ;@ PRINT

BL inc\_counter

LDR R9, =0x3 ;@ PRINT\_RET

BL inc\_counter

LDR R9, =0x8 ;@ RET\_POPPED

BL inc\_counter

LDR R9, =0xB ;@ NEW\_LINE

BL inc\_counter

LDR R9, =0xD ;@ VERIFY

BL inc\_counter

LDR R9, =0xE ;@ Extended opcode

BL inc\_counter

LDR R9, =0xF ;@ PIRACY

BL inc\_counter

LDR R9, =0xD ;@ VERIFY

BL inc\_counter

inc\_counter:

STR R9, [R0, R2]

ADD R2, R2, #1 ;@ increment Zmem offset

ADD R9, R9, #1

MOV R15, R14

no\_operands:

OP0\_0: ;@ RTRUE

OP0\_1: ;@ RFALSE

OP0\_2: ;@ PRINT

OP0\_3: ;@ PRINT\_RET

OP0\_8: ;@ RET\_POPPED

OP0\_B: ;@ NEW\_LINE

OP0\_D: ;@ VERIFY

OP0\_E: ;@ Extended opcode

one\_operand:

OP1\_00: ;@ JZ

OP1\_1: ;@ GET\_SIBLING

OP1\_2: ;@ GET\_CHILD

OP1\_3: ;@ GET\_PARENT

OP1\_5: ;@ INC

OP1\_6: ;@ DEC

OP1\_8: ;@ CALL\_IS

OP1\_B: ;@ RET

OP1\_D: ;@ PRINT\_PADDR

OP1\_F: ;@ CALL\_IN

two\_operands:

OP2\_01: ;@ JE

OP2\_02: ;@ JL

OP2\_03: ;@ JG

OP2\_06: ;@ JIN

OP2\_08: ;@ OR

OP2\_09: ;@ AND

OP2\_0A: ;@ TEST\_ATTR

OP2\_0B: ;@ SET\_ATTR

OP2\_0C: ;@ CLEAR\_ATTR

OP2\_0D: ;@ STORE

OP2\_14: ;@ ADD

OP2\_15: ;@ SUB

OP2\_16: ;@ MUL

OP2\_17: ;@ DIV

OP2\_18: ;@ MOD

OP2\_19: ;@ CALL\_2S

OP2\_1A: ;@ CALL\_2N

OP2\_1B: ;@ SET\_COLOUR

var\_operands:

VAR\_05: ;@ PRINT\_CHAR

VAR\_06: ;@ PRINT\_NUM

VAR\_07: ;@ RANDOM

VAR\_08: ;@ PUSH

VAR\_09: ;@ PULL

VAR\_11: ;@ SET\_TEXT\_STYLE

VAR\_14: ;@ INPUT\_STREAM

VAR\_18: ;@ NOT

Hex character = 4 bits

2 characters = 1 byte = 8 bits

**From Lab 6:**

asm\_main:

LDR R1, =0x41200000 ;@ address of the butttons

LDR R3, =memory ;@ memory space reserved

LDR R5, =0xE0001030 ;@ Memory location where hello world is stored

;@ Set up transmitter to what TeraTerm is expecting

LDR R9, =0x20

LDR R4, =0xE0001004 ;@ Mode Register -- Tera Term transmitter

STR R9, [R4, #0]

;@ Start Baud rate setup

LDR R9, =0x3E

LDR R4, =0xE0001018 ;@ Baud rate generator

STR R9, [R4, #0]

;@ Complete baud rate setup

LDR R9, =0x6

LDR R4, =0xE0001034 ;@ Baud rate divider

STR R9, [R4, #0]

;@ Enable and reset the UART

LDR R9, =0x117

LDR R4, =0xE0001000 ;@ Control register

STR R9, [R4, #0]

MOV R9, #0 ;@ free register for stuff

MOV R2, #4 ;@ Start the character counter at 4

LDR R8, =100000 ;@ Counter for how many times "Hello, World" was displayed on screen

Store\_Char\_Dmem:

LDR R12, =0x48 ;@ H

BL inc\_counter

LDR R12, =0x65 ;@ e

BL inc\_counter

LDR R12, =0x6C ;@ l

BL inc\_counter

LDR R12, =0x6C ;@ l

BL inc\_counter

LDR R12, =0x6F ;@ o

BL inc\_counter

LDR R12, =0x2C ;@ ,

BL inc\_counter

LDR R12, =0x20 ;@

BL inc\_counter

LDR R12, =0x57 ;@ W

BL inc\_counter

LDR R12, =0x6F ;@ o

BL inc\_counter

LDR R12, =0x72 ;@ r

BL inc\_counter

LDR R12, =0x6C ;@ l

BL inc\_counter

LDR R12, =0x64 ;@ d

STR R12, [R3, R2]

MOV R2, #4

load\_store\_inc\_counter:

CMP R8, #0

BEQ done

LDR R12, [R3, R2]

STR R12, [R5, #0]

ADD R2, R2, #4

check\_buffer:

LDR R4, =0xE000102C

LDR R4, [R4, #0]

LDR R10, =8

AND R10, R4, R10

CMP R10, #8 ;@ Check 3rd bit to see if Transmitter FIFO is empty

BEQ NewLine\_Return

BNE check\_buffer

done:

B done

/\* For Button Use (Part 2)

CHKBTN:

LDR R4, [R1, #0]

MOV R11, R4

CMP R4, #0

BEQ CHKBTN

BL delay\_loop

BTN\_Change:

LDR R4, [R1, #0]

CMP R4, R11

BEQ BTN\_Change

BL delay\_loop

display\_shift:

LDR R12, [R3, R2]

STR R12, [R5, #0]

ADD R2, R2, #4

CMP R2, #52

BLEQ NewLine\_Return

B Store\_Char

\*/

NewLine\_Return:

CMP R2, #52

BNE load\_store\_inc\_counter

LDR R9, =0xA

STR R9, [R5, #0]

LDR R9, =0xD

STR R9, [R5, #0]

MOV R2, #4

SUB R8, R8, #1

B load\_store\_inc\_counter

inc\_counter:

STR R12, [R3, R2]

ADD R2, R2, #4

MOV R15, R14

/\* For Button Use (Part 2)

delay\_loop:

LDR R10, =500000

loop:

SUBS R10, R10, #1

BNE loop

MOV R15, R14

\*/

=====================================================================================================================================================================================================================================================================================================================================

.global asm\_main

.align 4

memory: .space 2000000 ;@ 2MB reserved for own use throughout program

Zmem: .space 2000000 ;@ 2MB for uploading Zprogram

Zstack: .space 1000000 ;@ 1MB for Zstack

FZreg: .space 1000000 ;@ 1MB for function local Zregisters

temp: .space 256 ;@ 256B temporary storage for holding temp operands to variable operand count instructions

.align 4

asm\_main:

;@ Set up transmitter to what TeraTerm is expecting

LDR R9, =0x20

LDR R4, =0xE0001004 ;@ Mode Register -- Tera Term transmitter

STR R9, [R4, #0]

;@ Start Baud rate setup

LDR R9, =0x3E

LDR R4, =0xE0001018 ;@ Baud rate generator

STR R9, [R4, #0]

;@ Complete baud rate setup

LDR R9, =6

LDR R4, =0xE0001034 ;@ Baud rate divider

STR R9, [R4, #0]

;@ Enable and reset the UART

LDR R9, =0x117

LDR R4, =0xE0001000 ;@ Control register

STR R9, [R4, #0]

;@ address of the sliding switches

LDR R1, =0x41220000

;@ read initial value of sliders

LDR R8, [R1, #0]

LDR R11, =memory

STR R8, [R11, #0] ;@ At the zero-th position in memory we store the current value of the sliders

LDR R7, =Zmem ;@ FIXED VALUE DON'T CHANGE R7

LDR R10, =0 ;@ initial value of the receiver fifo store counter

STR R10, [R7, #0] ;@ At the fourth position in memory we store the current value of upload pointer

Check\_switch:

LDR R1, =0x41220000 ;@ address of the sliding switches

LDR R8, [R11, #0] ;@ Getting the value of last state of sliders

LDR R10, =104800

LDR R9, [R1, #0]

CMP R9, R8

BEQ Check\_switch

BNE Delay

Delay:

SUBS R10, R10, #1

BEQ Verify\_switch

BNE Delay

Verify\_switch:

LDR R9, [R1, #0]

LDR R10, =104800

CMP R8, R9

BEQ Check\_switch

BNE Switch\_changed

Switch\_changed:

MOV R8, R9

LDR R11, =memory

STR R8, [R11, #0]

LDR R9, =128

AND R9, R8, R9

CMP R9, #128

BLEQ upload\_mode

LDR R9, =64

AND R9, R8, R9

CMP R9, #64

BLEQ no\_header\_mode

BLNE header\_mode

LDR R9, =128

AND R9, R8, R9

CMP R9, #128

BLNE run\_mode

LDR R9, =32

AND R9, R8, R9

CMP R9, #32

BLEQ debug\_mode

BLNE game\_mode

B Check\_switch

upload\_mode:

PUSH {R8-R12, LR}

LDR R10, =Zmem

LDR R11, =2000000 ;@ Size of Zmem

LDR R8, =0 ;@ Zmem offset incrementor

LDR R9, =0 ;@ the null value to store at Zmem offset by R8

LDR R12, =1000000 ;@ Size of Zstack

BL Delete\_Zmemory

BL Reset\_Zstack

BL Check\_receiver\_fifo\_and\_store

BL Check\_transmitter\_fifo\_and\_print

POP {R8-R12, PC}

;@MOV PC, LR

\_done:

B \_done

;@ BL Check\_transmitter\_fifo\_and\_

Check\_receiver\_fifo\_and\_store:

LDR R11, =2 ;@ 2

LDR R8, =0xE0001030 ;@ Memory location where hello world is stored

LDR R9, =0xE000102C ;@ XUARTPS\_SR\_OFFSET

LDR R9, [R9, #0]

;@TST R9, #0b00000010

AND R11, R9, R11

CMP R11, #2 ;@ Check 2nd bit to see if receiver FIFO is empty

BEQ Store

MOVNE PC, LR

;@BNE Check\_receiver\_fifo\_and\_store

Store:

LDR R9, [R7, #0] ;@ Loading the value of the fifo receiver store counter to R9

LDRB R8, [R8, #0] ;@ loading the input from UART memory location to R8

ADD R9, R9, #4

STRB R8, [R7, R9] ;@ Storing

STR R9, [R7, #0] ;@ Update the value of the fifo receiver store counter

B Check\_receiver\_fifo\_and\_store

Check\_transmitter\_fifo\_and\_print:

;@LDR R11, =16384 ;@ 2^14

LDR R11, =8 ;@ 8

;@LDR R8, =0xE0001030 ;@ Memory location where hello world is stored

LDR R9, =0xE000102C ;@ XUARTPS\_SR\_OFFSET

LDR R9, [R9, #0]

;@TST R9, R11

AND R11, R9, R11

;@CMP R11, #16384 ;@ Check 14th bit to see if Transmitter FIFO is empty

CMP R11, #8

LDREQ R11, =4

BEQ print

BNE Check\_transmitter\_fifo\_and\_print

print:

LDR R9, [R7, #0]

LDR R10, [R7, R11]

ADD R11, R11, #4

CMP R11, R9

MOVEQ PC, LR

STR R10, [R8, #0]

Delete\_Zmemory:

STR R9, [R10, R8]

ADD R8, R8, #4

CMP R8, R11

BNE Delete\_Zmemory

LDREQ R8, =0

MOVEQ PC, LR

Reset\_Zstack:

STR R9, [R10, R8]

ADD R8, R8, #4

CMP R8, R12

BNE Delete\_Zmemory

LDREQ R8, =0

MOVEQ PC, LR

run\_mode:

MOV PC, LR

no\_header\_mode:

MOV PC, LR

header\_mode:

MOV PC, LR

debug\_mode:

MOV PC, LR

game\_mode:

MOV PC, LR

/\*

check\_receiver\_buffer\_and\_print:

LDR R4, =0xE000102C

LDR R4, [R4, #0]

LDR R10, =2

AND R10, R4, R10

CMP R10, #2 ;@ Check 1st bit to see if Receiver FIFO is empty

BEQ check\_receiver\_buffer\_and\_print

STR R2, [R5, #0]

B Check\_switch

done:

B done

NewLine\_Return:

CMP R2, #52

BNE load\_store\_inc\_counter

LDR R9, =0xA

STR R9, [R5, #0]

LDR R9, =0xD

STR R9, [R5, #0]

MOV R2, #4

SUB R8, R8, #1

B load\_store\_inc\_counter

inc\_counter:

STR R12, [R3, R2]

ADD R2, R2, #4

MOV R15, R14

\*/

/\*

Store\_Char\_Dmem:

LDR R12, =0x48 ;@ H

BL inc\_counter

LDR R12, =0x65 ;@ e

BL inc\_counter

LDR R12, =0x6C ;@ l

BL inc\_counter

LDR R12, =0x6C ;@ l

BL inc\_counter

LDR R12, =0x6F ;@ o

BL inc\_counter

LDR R12, =0x2C ;@ ,

BL inc\_counter

LDR R12, =0x20 ;@

BL inc\_counter

LDR R12, =0x57 ;@ W

BL inc\_counter

LDR R12, =0x6F ;@ o

BL inc\_counter

LDR R12, =0x72 ;@ r

BL inc\_counter

LDR R12, =0x6C ;@ l

BL inc\_counter

LDR R12, =0x64 ;@ d

STR R12, [R3, R2]

MOV R2, #4

load\_store\_inc\_counter:

CMP R8, #0

BEQ done

LDR R12, [R3, R2]

STR R12, [R5, #0]

ADD R2, R2, #4

\*/

/\* For Button Use (Part 2)

delay\_loop:

LDR R10, =500000

loop:

SUBS R10, R10, #1

BNE loop

MOV R15, R14

For Button Use (Part 2)

CHKBTN:

LDR R4, [R1, #0]

MOV R11, R4

CMP R4, #0

BEQ CHKBTN

BL delay\_loop

BTN\_Change:

LDR R4, [R1, #0]

CMP R4, R11

BEQ BTN\_Change

BL delay\_loop

display\_shift:

LDR R12, [R3, R2]

STR R12, [R5, #0]

ADD R2, R2, #4

CMP R2, #52

BLEQ NewLine\_Return

B Store\_Char

\*/