```
################################################################################
# Name:    Samuel Wait
# Class:   CS2318-253 (Assembly Language, Spring 2023)
# Subject: Assignment 3 Part 1
# Date:    4/30/23
################################################################################
# MIPS assembly language translation of a given C++ program that, except for the
# main function, involves "trivial" functions each of which:
# - is a leaf function
# - does not require local storage (on the stack)
# NOTES:
# - "does not require local storage" means each (leaf) function
#   -- does not need memory on the stack for local variables (including arrays)
#   -- WILL NOT use any callee-saved registers ($s0 through $s7)
# - meant as an exercise for familiarizing w/ the
#   -- basics of MIPS' function-call mechanism
#   -- how-to's of pass-by-value & pass-by-address when doing functions in MIPS
# - does NOT adhere to yet-to-be-studied function-call convention (which is
#   needed when doing functions in general, not just "trivial" functions)
# - main (being the only non-"trivial" function & an unavoidable one) will in
#   fact violate the yet-to-be-studied function-call convention
#   -- due to this, each of the functions that main calls MUST TAKE ANOMALOUS
#      CARE not to "clobber" the contents of registers that main uses & expects
#      to be preserved across calls
#   -- experiencing the pains and appreciating the undesirability of having to
#      deal with the ANOMALOUS SITUATION (due to the non-observance of any
#      function-call convention that governs caller-callee relationship) should
#      help in understanding why some function-call convention must be defined
#      and observed
################################################################################
# Algorithm used:
# Given C++ program (Assign03P1.cpp)
################################################################################
# Sample test run:
##################
#
# vals to do? 4
# enter an int: 1
# enter an int: 2
# enter an int: 3
# enter an int: 4
# original:
# 1 2 3 4
# backward:
# 4 3 2 1
# do more? y
```

```
# vals to do? 0
# 0 is bad, make it 1
# enter an int: 5
# original:
# 5
# backward:
# 5
# do more? y
# vals to do? 8
# 8 is bad, make it 7
# enter an int: 7
# enter an int: 6
# enter an int: 5
# enter an int: 4
# enter an int: 3
# enter an int: 2
# enter an int: 1
# original:
# 7 6 5 4 3 2 1
# backward:
# 1 2 3 4 5 6 7
# do more? n
# -- program is finished running --
################################################################################
# int GetOneIntByVal(const char vtdPrompt[]);
# void GetOneIntByAddr(int* intVarToPutInPtr,const char prompt[]);
# void GetOneCharByAddr(char* charVarToPutInPtr, const char prompt[]);
# void ValidateInt(int* givenIntPtr, int minInt, int maxInt, const char msg[]);
# void SwapTwoInts(int* intPtr1, int* intPtr2);
# void ShowIntArray(const int array[], int size, const char label[]);
#
#int main()
#{
                                                                                .text
                                                                                .globl main
main:
#   int intArr[7];
#   int valsToDo;
#   char reply;
#   char vtdPrompt[] = "vals to do? ";
#   char entIntPrompt[] = "enter an int: ";
#   char adjMsg[] = " is bad, make it ";
#   char origLab[] = "original:\n";
#   char backLab[] = "backward:\n";
#   char dmPrompt[] = "do more? ";
#   int i, j;
```

```
#################

# Register Usage:

#################

# $t0: register holder for a value

# $t1: i

# $t2: j

#################

                                                                addiu $sp, $sp, -113

                                                                j StrInitCode               # clutter-reduction jump (string initialization)

endStrInit:

#   do

#   {

begWBodyM1:

                                                                li $a0, '\n'

                                                                li $v0, 11

                                                                syscall                     # '\n' to offset effects of syscall #12 drawback

#       valsToDo = GetOneIntByVal(vtdPrompt);
###################(3)####################

                                                                addi $a0, $sp, 100
                                                                jal GetOneIntByVal
                                                                sw $v0, 85($sp)


#       ValidateInt(&valsToDo, 1, 7, adjMsg);
###################(4)####################

                                                                addi $a0, $sp, 85
                                                                li $a1, 1
                                                                li $a2, 7
                                                                addi $a3, $sp, 3

                                                                jal ValidateInt

#       for (i = valsToDo; i > 0; --i)
###################(1)####################

                                                                lw $t1, 85($sp)

                                                                j FTestM1
begFBodyM1:

#           if (i % 2) // i is odd

                                                                andi $t0, $t1, 0x00000001

                                                                beqz $t0, ElseI1

#               intArr[valsToDo - i] = GetOneIntByVal(entIntPrompt);
###################(8)####################

                                                                addi $a0, $sp, 42
                                                                jal GetOneIntByVal
                                                                lw $t3, 85($sp)
                                                                sub $t0, $t0, $t1
                                                                sll $t0, $t0, 2
                                                                add $t0, $sp, $t0
                                                                sw $v0, 57($t0)

                                                                j endI1

#           else // i is even

ElseI1:

#               GetOneIntByAddr(intArr + valsToDo - i, entIntPrompt);
###################(7)####################

                                                                lw $t0, 85($sp)
                                                                sub $t0, $t0, $t1
                                                                sll $t0, $t0, 2
                                                                addi $a0, $t0, 57
                                                                add $a0, $a0, $sp
                                                                addi $a1, $sp, 42
                                                                jal GetOneIntByAddr
```

```
endI1:
                                                                addi $t1, $t1, -1

FTestM1:
                                                                bgtz $t1, begFBodyM1

#       ShowIntArray(intArr, valsToDo, origLab);
####################(3)####################
                                                                addi $a0, $sp, 57
                                                                lw $a1, 85($sp)
                                                                addi $a2, $sp, 91

                                                                jal ShowIntArray

#       for (i = 0, j = valsToDo - 1; i < j; ++i, --j)
####################(3)####################
                                                                li $t1, 0
                                                                lw $t2, 85($sp)
                                                                addi $t2, $t2, -1

                                                                j FTestM2

begFBodyM2:

#           SwapTwoInts(intArr + i, intArr + j);
####################(5)####################
                                                                addi $a0, $sp, 57
                                                                sll $t0, $t1, 2
                                                                add $a0, $a0, $t0
                                                                addi $a1, $sp, 57
                                                                sll $t0, $t2, 2
                                                                add $a1, $a1, $t0

                                                                jal SwapTwoInts

                                                                addi $t1, $t1, 1

                                                                addi $t2, $t2, -1

FTestM2:
                                                                blt $t1, $t2, begFBodyM2

#       ShowIntArray(intArr, valsToDo, backLab);
####################(3)####################
                                                                addi $a0, $sp, 57
                                                                lw $a1, 85($sp)
                                                                addi $a2, $sp 31

                                                                jal ShowIntArray

#       GetOneCharByAddr(&reply, dmPrompt);
####################(2)####################
                                                                addi $a0, $sp, 1
                                                                addi $a1, $sp, 21

                                                                jal GetOneCharByAddr

#   }

#   while (reply != 'n' && reply != 'N');
####################(1)####################
                                                                lb $v1, 76($sp)

                                                                li $t0, 'n'

                                                                beq $v1, $t0, endWhileM1

                                                                li $t0, 'N'

                                                                bne $v1, $t0, begWBodyM1

endWhileM1:                                                     # extra helper label added
#   return 0;

#}
                                                                addiu $sp, $sp, 113

                                                                li $v0, 10

                                                                syscall

################################################################################
```

```
#int GetOneIntByVal(const char prompt[])
#{
GetOneIntByVal:
#   int oneInt;
#   cout << prompt;
                                                                    li $v0, 4
                                                                    syscall
#   cin >> oneInt;
                                                                    li $v0, 5
                                                                    syscall
#   return oneInt;
#}
                                                                    jr $ra


################################################################################
#void GetOneIntByAddr(int* intVarToPutInPtr, const char prompt[])
#{
GetOneIntByAddr:
#   cout << prompt;
                                                                    move $t0, $a0               # $t0 has saved copy of $a0 as received
                                                                    move $a0, $a1
                                                                    li $v0, 4
                                                                    syscall
#   cin >> *intVarToPutInPtr;
                                                                    li $v0, 5
                                                                    syscall
                                                                    sw $v0, 0($t0)
#}
                                                                    jr $ra


################################################################################
#void ValidateInt(int* givenIntPtr, int minInt, int maxInt, const char msg[])
#{
ValidateInt:
#################
# Register Usage:
#################
# $t0: copy of arg1 ($a0) as received
# $v1: value loaded from mem (*givenIntPtr)
#################
                                                                    move $t0, $a0               # $t0 has saved copy of $a0 as received
#   if (*givenIntPtr < minInt)
#   {
                                                                    lw $v1, 0($t0)              # $v1 has *givenIntPtr
                                                                    bge $v1, $a1, ElseVI1
#      cout << *givenIntPtr << msg << minInt << endl;
```

```
                                                                        move $a0, $v1
                                                                        li $v0, 1
                                                                        syscall
                                                                        move $a0, $a3
                                                                        li $v0, 4
                                                                        syscall
                                                                        move $a0, $a1
                                                                        li $v0, 1
                                                                        syscall
                                                                        li $a0, '\n'
                                                                        li $v0, 11
                                                                        syscall
#       *givenIntPtr = minInt;
                                                                        sw $a1, 0($t0)
                                                                        j endIfVI1
#   }
#   else
#   {
ElseVI1:
#       if (*givenIntPtr > maxInt)
#       {
                                                                        ble $v1, $a2, endIfVI2
#           cout << *givenIntPtr << msg << maxInt << endl;
                                                                        move $a0, $v1
                                                                        li $v0, 1
                                                                        syscall
                                                                        move $a0, $a3
                                                                        li $v0, 4
                                                                        syscall
                                                                        move $a0, $a2
                                                                        li $v0, 1
                                                                        syscall
                                                                        li $a0, '\n'
                                                                        li $v0, 11
                                                                        syscall
#           *givenIntPtr = maxInt;
                                                                        sw $a2, 0($t0)
#       }
endIfVI2:
#   }
endIfVI1:
#}
                                                                        jr $ra

################################################################################
#void ShowIntArray(const int array[], int size, const char label[])
```

```
#{
ShowIntArray:
#################
# Register Usage:
#################
# $t0: copy of arg1 ($a0) as received
# $a3: k
# $v1: value loaded from mem (*givenIntPtr)
#################
                                                    move $t0, $a0              # $t0 has saved copy of $a0 as received
#   cout << label;
                                                    move $a0, $a2
                                                    li $v0, 4
                                                    syscall
#   int k = size;
                                                    move $a3, $a1
                                                    j WTestSIA
#   while (k > 0)
#   {
begWBodySIA:
#      cout << array[size - k] << ' ';
                                                    sub $v1, $a1, $a3          # $v1 gets (size - k)
                                                    sll $v1, $v1, 2            # $v1 now has 4*(size - k)
                                                    add $v1, $v1, $t0          # $v1 now has &array[size - k]
                                                    lw $a0, 0($v1)             # $a0 has array[size - k]
                                                    li $v0, 1
                                                    syscall
                                                    li $a0, ' '
                                                    li $v0, 11
                                                    syscall
#      --k;
                                                    addi $a3, $a3, -1
#   }
WTestSIA:
                                                    bgtz $a3, begWBodySIA
#   cout << endl;
                                                    li $a0, '\n'
                                                    li $v0, 11
                                                    syscall
#}
                                                    jr $ra

################################################################################
#void SwapTwoInts(int* intPtr1, int* intPtr2)
#{
SwapTwoInts:
```

```
#################

# Register Usage:

#################

# (fill in where applicable)

#################

#   int temp = *intPtr1;

#  *intPtr1 = *intPtr2;

#   *intPtr2 = temp;



###################(4)###################

                                                                                lw $t0, 0($a0)
                                                                                lw $t3, 0($a1)
                                                                                sw $t3, 0($a0)
                                                                                sw $t0, 0($a1)

#

                                                                                jr $ra



##############################################################################

#void GetOneCharByAddr(char* charVarToPutInPtr, const char prompt[])

#{

GetOneCharByAddr:

#################

# Register Usage:

#################

# (fill in where applicable)

#################

#   cout << prompt;

#   cin >> *charVarToPutInPtr;



###################(7)###################

                                                                                move $t0, $a0
                                                                                move $a0, $a1
                                                                                li $v0, 4
                                                                                syscall
                                                                                li $v0, 12
                                                                                syscall
                                                                                sb $v0, 0($t0)

#}

                                                                                jr $ra



##############################################################################

StrInitCode:

#################

# "bulky & boring" string-initializing code move off of main stage

##############################################################################

                                                                                li $t0, ' '

                                                                                sb $t0, 3($sp)

                                                                                li $t0, 'i'

                                                                                sb $t0, 4($sp)

                                                                                li $t0, 's'

                                                                                sb $t0, 5($sp)
```

```
li $t0, ' '
sb $t0, 6($sp)
li $t0, 'b'
sb $t0, 7($sp)
li $t0, 'a'
sb $t0, 8($sp)
li $t0, 'd'
sb $t0, 9($sp)
li $t0, ','
sb $t0, 10($sp)
li $t0, ' '
sb $t0, 11($sp)
li $t0, 'm'
sb $t0, 12($sp)
li $t0, 'a'
sb $t0, 13($sp)
li $t0, 'k'
sb $t0, 14($sp)
li $t0, 'e'
sb $t0, 15($sp)
li $t0, ' '
sb $t0, 16($sp)
li $t0, 'i'
sb $t0, 17($sp)
li $t0, 't'
sb $t0, 18($sp)
li $t0, ' '
sb $t0, 19($sp)
li $t0, '\0'
sb $t0, 20($sp)
```

##############

```
li $t0, 'o'
sb $t0, 89($sp)
li $t0, 'r'
sb $t0, 90($sp)
li $t0, 'i'
sb $t0, 91($sp)
li $t0, 'g'
sb $t0, 92($sp)
li $t0, 'i'
sb $t0, 93($sp)
li $t0, 'n'
sb $t0, 94($sp)
li $t0, 'a'
sb $t0, 95($sp)
li $t0, 'l'
```

```
                                                    sb $t0, 96($sp)
                                                    li $t0, ':'
                                                    sb $t0, 97($sp)
                                                    li $t0, '\n'
                                                    sb $t0, 98($sp)
                                                    li $t0, '\0'
                                                    sb $t0, 99($sp)
#############
                                                    li $t0, 'v'
                                                    sb $t0, 100($sp)
                                                    li $t0, 'a'
                                                    sb $t0, 101($sp)
                                                    li $t0, 'l'
                                                    sb $t0, 102($sp)
                                                    li $t0, 's'
                                                    sb $t0, 103($sp)
                                                    li $t0, ' '
                                                    sb $t0, 104($sp)
                                                    li $t0, 't'
                                                    sb $t0, 105($sp)
                                                    li $t0, 'o'
                                                    sb $t0, 106($sp)
                                                    li $t0, ' '
                                                    sb $t0, 107($sp)
                                                    li $t0, 'd'
                                                    sb $t0, 108($sp)
                                                    li $t0, 'o'
                                                    sb $t0, 109($sp)
                                                    li $t0, '?'
                                                    sb $t0, 110($sp)
                                                    li $t0, ' '
                                                    sb $t0, 111($sp)
                                                    li $t0, '\0'
                                                    sb $t0, 112($sp)
#############
                                                    li $t0, 'd'
                                                    sb $t0, 21($sp)
                                                    li $t0, 'o'
                                                    sb $t0, 22($sp)
                                                    li $t0, ' '
                                                    sb $t0, 23($sp)
                                                    li $t0, 'm'
                                                    sb $t0, 24($sp)
                                                    li $t0, 'o'
                                                    sb $t0, 25($sp)
                                                    li $t0, 'r'
```

```
                                                    sb $t0, 26($sp)
                                                    li $t0, 'e'
                                                    sb $t0, 27($sp)
                                                    li $t0, '?'
                                                    sb $t0, 28($sp)
                                                    li $t0, ' '
                                                    sb $t0, 29($sp)
                                                    li $t0, '\0'
                                                    sb $t0, 30($sp)
##############
                                                    li $t0, 'e'
                                                    sb $t0, 42($sp)
                                                    li $t0, 'n'
                                                    sb $t0, 43($sp)
                                                    li $t0, 't'
                                                    sb $t0, 44($sp)
                                                    li $t0, 'e'
                                                    sb $t0, 45($sp)
                                                    li $t0, 'r'
                                                    sb $t0, 46($sp)
                                                    li $t0, ' '
                                                    sb $t0, 47($sp)
                                                    li $t0, 'a'
                                                    sb $t0, 48($sp)
                                                    li $t0, 'n'
                                                    sb $t0, 49($sp)
                                                    li $t0, ' '
                                                    sb $t0, 50($sp)
                                                    li $t0, 'i'
                                                    sb $t0, 51($sp)
                                                    li $t0, 'n'
                                                    sb $t0, 52($sp)
                                                    li $t0, 't'
                                                    sb $t0, 53($sp)
                                                    li $t0, ':'
                                                    sb $t0, 54($sp)
                                                    li $t0, ' '
                                                    sb $t0, 55($sp)
                                                    li $t0, '\0'
                                                    sb $t0, 56($sp)
##############
                                                    li $t0, 'b'
                                                    sb $t0, 31($sp)
                                                    li $t0, 'a'
                                                    sb $t0, 32($sp)
                                                    li $t0, 'c'
```

```
sb $t0, 33($sp)

li $t0, 'k'

sb $t0, 34($sp)

li $t0, 'w'

sb $t0, 35($sp)

li $t0, 'a'

sb $t0, 36($sp)

li $t0, 'r'

sb $t0, 37($sp)

li $t0, 'd'

sb $t0, 38($sp)

li $t0, ':'

sb $t0, 39($sp)

li $t0, '\n'

sb $t0, 40($sp)

li $t0, '\0'

sb $t0, 41($sp)


j endStrInit
```