Project 5 - MIPS Programs with Memory-Mapped I/O - amfahe25 Aidan Fahey

collaboration.txt

```
I collaborated with Charlie but I didn't actually help him he just helped me
```

scream.txt

```
.text
LUI $2, 0x8000 // sets register 2 to address
LI $3, 15 // for loop limit
LI $4, 64 // value to store at m[8]
LI $5, 97 // value to store at m[8]
LI $6, 33 // value to store at m[8]
LI $7, 32 // value to store at m[8]

main:
SB $4, 16($2) // sets m[8] to 64

loop:
BGT $2, $3, exit // is the loop
SB $5, 16($2) // sets m[8] to 97
ADDIU $2, $0, 1 // increments for loop
J loop // jumps to top of loop

exit: // after loop
SB $6, 16($2) // sets m[8] to 33
SB $7, 16($2) // sets m[8] to 32
```

`Assembling scream.txt ...`

Reading assembly code from file: scream.S

Performing first pass: determining memory layout and addresses.

Performing second pass: encoding instructions and data.

Finished processing. Writing out code and data.

Writing assembled code to file: scream_code.txt

SUCCESS! No errors or warnings

Simulate scream with no keyboard input

```
Loading logisim-style memory image 'scream_code.txt' ...
Executing MIPS code...
MIPS program output will be in PLAIN TEXT.
Simulator messages will be in PLAIN TEXT as well.
Note: Each character of user input is shown in curly-braces, e.g. {H}{i}{!}{\n}

MIPS code finishes with result $v0 = 0x80000000 (decimal -2147483648, unsigned 2147483648).
MIPS processor executed approx. 12 instructions in 1478 nsec at 8.8119 MHz.
Final state of registers:
    $0 = 0x00000000    $t0 = 0x00000000    $s0 = 0x00000000    $t8 = 0x00000000
   $at = 0x7ffffff1    $t1 = 0x00000000    $s1 = 0x00000000    $t9 = 0x00000000
   $v0 = 0x80000000    $t2 = 0x00000000    $s2 = 0x00000000    $k0 = 0x00000000
   $v1 = 0x0000000f    $t3 = 0x00000000    $s3 = 0x00000000    $k1 = 0x00000000
   $a0 = 0x00000040    $t4 = 0x00000000    $s4 = 0x00000000    $gp = 0x00000000
   $a1 = 0x00000061    $t5 = 0x00000000    $s5 = 0x00000000    $sp = 0x00000000
   $a2 = 0x00000021    $t6 = 0x00000000    $s6 = 0x00000000    $fp = 0x00000000
   $a3 = 0x00000020    $t7 = 0x00000000    $s7 = 0x00000000    $ra = 0xfffffffc
   $hi = 0x00000000    $pc = 0x00000038
   $lo = 0x00000000   $npc = 0x0000003c
All data memory locations written by program:
```

translate.txt

```
.text
LUI $8, 0x8000   # sets m = 0x80000000, our base address for input/output

waitloop:
LB $9, 0($8)     # reads a byte from m[0], to query the keyboard status
BEQ $9, $0, waitloop

LB $10, 4($8)    # reads a byte from m[1], to query the keyboard data

ADDIU $10, $10, -48    # decrement r10 by 48, which is the ascii code for '0'
ADDIU $10, $10, 97    # increment r10 by 97, which is the ascii code for 'a'
LB $10, 16($8)        # store r10 as a single byte into memory address 0x80000008
J waitloop            # go back to the waitloop near the top
```

Assembling translate.txt ...

Reading assembly code from file: translate.S

Performing first pass: determining memory layout and addresses.

Performing second pass: encoding instructions and data.

Finished processing. Writing out code and data.

Writing assembled code to file: translate_code.txt

SUCCESS! No errors or warnings

Simulate translate with input [0123456789]

```
Loading logisim-style memory image 'translate_code.txt' ...
Executing MIPS code...
MIPS program output will be in PLAIN TEXT.
Simulator messages will be in PLAIN TEXT as well.
Note: Each character of user input is shown in curly-braces, e.g. {H}{i}{!}{\n}
{0}{1}{2}{3}{4}{5}{6}{7}{8}{9}
MIPS processor has run for 200000 cycles, but not yet halted.
Processor has executed approx. 200000 instructions in 26689049 nsec at 7.7493 MHz.
MIPS processor choked at pc 0x00000008.
Last 8 instructions executed and current instruction:
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
==> 0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
```

Simulate translate with input [27834]

```
Loading logisim-style memory image 'translate_code.txt' ...
Executing MIPS code...
MIPS program output will be in PLAIN TEXT.
Simulator messages will be in PLAIN TEXT as well.
Note: Each character of user input is shown in curly-braces, e.g. {H}{i}{!}{\n}
{2}{7}{8}{3}{4}
MIPS processor has run for 200000 cycles, but not yet halted.
Processor has executed approx. 200000 instructions in 26882530 nsec at 7.7439 MHz.
MIPS processor choked at pc 0x00000004.
Last 8 instructions executed and current instruction:
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
    0x00000004: 81090000: lb $t1, 0($t0)
    0x00000008: 1120fffe: beq $t1, $0, -2  # address 0x00000004
==> 0x00000004: 81090000: lb $t1, 0($t0)
```

sort.txt

```
.data

numvals: .word 6     # this memory holds an integer, the array size

vals:    .word 99    # these next values are the array contents
         .word 15
         .word 1044942
         .word -5
         .word 35
         .word 0x8BADF00D  # note: this is a negative number

msg1:    .asciz "Sorting..."  # a zero-terminated string

msg2:    .asciz "Done!\n"     # another zero-terminated string

.text

# initialization
LUI $8, 0x8000         # sets m = 0x80000000, our base address for input/output

# print msg1
LA $4, msg1            # r4 = address of msg1 in memory
loop1:
LB $7, 0($4)           # get next ascii byte of msg1
BEQ $7, $0, end        # break out of this loop when end of msg1 is reached
SB $4, 8($8)           # print this ascii byte
ADDIU $4, $4, 1
J loop1                # go back to top of loop1

end1:
nop

# sort array
main:

LA $2, numvals # r2 = address of numvals variable in memory
LW $2, 0($2)  # r2 = get value (4 bytes) of numvals variable from memory

mainLoop:
ADDIU $3, $2, -1 # count for pass
```

```
BLEZ $3, mainDone # done main

LA $4, vals # address of array
LI $5, 0 # did swap check

JAL loopPass # single sort

BEQ $5, $0, mainDone # if no swaps, done

ADDIU $2, $2, -1 # decrement remaining passes
BEQ $0, $0, main

mainDone:
J end # done life

loopPass:
LW $6, 0($4) # load first element of array in $6
LW $7, 4($4) # load first element of array in $7
BGT $6, $7, swapPass # if $6 > $7, swap

passNext:
ADDIU $4, $4, 4 # increment pointer to next index
ADDIU $3, $3, -1 # decrement number of loops
BGTZ $3, loopPass # loop if not swap not passed
JR $RA # return to call

swapPass:
SW $6, 4 ($4) # store [i+1] in $6
SW $7, 0 ($4) # store [i] in $7
LI $5, 1 # tell main loop a swap happened
J passNext

end:
NOP # cry

# print msg2
LA $4, msg2           # r4 = address of msg2 in memory
loop2:
LB $7, 0($4)          # get next ascii byte of msg1
BEQ $7, $0, end2      # break out of this loop when end of msg1 is reached
SB $4, 8($8)          # print this ascii byte
ADDIU $4, $4, 1       # increment register to get next ascii byte
J loop2               # jump to top of loop

end2:
NOP
```

```
Assembling sort.txt ...
```

Reading assembly code from file: sort.S

Performing first pass: determining memory layout and addresses.

Performing second pass: encoding instructions and data.

Finished processing. Writing out code and data.

Writing assembled code to file: sort_code.txt

Writing assembled data to file: sort_data.txt

SUCCESS! No errors or warnings

Simulate sort with no keyboard input

```
Loading logisim-style memory image 'sort_code.txt' ...
Loading logisim-style memory image 'sort_data.txt' ...
Executing MIPS code...
MIPS program output will be in PLAIN TEXT.
Simulator messages will be in PLAIN TEXT as well.
Note: Each character of user input is shown in curly-braces, e.g. {H}{i}{!}{\n}
<<FS>><<GS>><<RS>><<US>> !"#$%'()*+,
MIPS code finishes with result $v0 = 0x00000000 (decimal 0, unsigned 0, char '\0').
MIPS processor executed approx. 92 instructions in 16550 nsec at 5.5558 MHz.
Final state of registers:
    $0 = 0x00000000    $t0 = 0x80000000    $s0 = 0x00000000    $t8 = 0x00000000
   $at = 0x00000000    $t1 = 0x00000000    $s1 = 0x00000000    $t9 = 0x00000000
   $v0 = 0x00000000    $t2 = 0x00000000    $s2 = 0x00000000    $k0 = 0x00000000
   $v1 = 0x00000000    $t3 = 0x00000000    $s3 = 0x00000000    $k1 = 0x00000000
   $a0 = 0x0000002d    $t4 = 0x00000000    $s4 = 0x00000000    $gp = 0x00000000
   $a1 = 0x00000000    $t5 = 0x00000000    $s5 = 0x00000000    $sp = 0x00000000
   $a2 = 0x00000000    $t6 = 0x00000000    $s6 = 0x00000000    $fp = 0x00000000
   $a3 = 0x00000000    $t7 = 0x00000000    $s7 = 0x00000000    $ra = 0xfffffffc
   $hi = 0x00000000    $pc = 0x000000ac
   $lo = 0x00000000   $npc = 0x000000b0
All data memory locations written by program:
```