Concurrency and Co-Design Lab 4
Pelagia Majoni, Ha Phan, Minor Reedich, Olga Shevchuk, Jacob Valero, Adrian Velonis
Logisim file

ReadMe:

1. Put string into RAM:
    a. First translate the input string (e.g. “Hello”) into ASCII code
    b. Copy and paste the ASCII code (in hexadecimal) into the RAM starting at C001
    c. The first spot (at C001) is for the number of characters in the input string (e.g. 5).The rest are for the ASCII code, for each character of the string. e.g. `5 48 65 6c 6c 6f`, starting at C001
2. Put HERA assembly code into ROM starting at 0000 (see bottom of document)
3. Before turning on the clock (really, before the random computation is done: `SET(R7, 20)`), type something into Keyboard 0 and Keyboard 1
4. Start ticking the clock if not already done (Simulate -> Tick Frequency -> Ticks Enabled).
    a. Step 3 output: input from keyboard 0 will immediately be printed out to screen 0.
5. After the text in Keyboard 1 is put into RAM (once the return character/Enter key is used), but *before* the text in Keyboard 1 is printed: type new strings into Keyboard 0 and Keyboard 1 (see Step 5):
    a. Step 4 output: Keyboard 1 will then print text stored in RAM to Screen 1, but with all lower-case letters converted to upper-case
    b. Step 4 output: Keyboard 1 will print the text stored in RAM to Screen 0, with non-alphanumeric characters replaced by underscores
6. The program then enters a loop in which it checks each keyboard to see if it has a character, and if so, prints that character to both screens:
    a. All letters from Keyboard 0 display in upper-case on both terminals
    b. All letters from Keyboard 1 display in lower-case on both terminals

---

OPCODE USAGE:

OPCODE(0x5000): print R1 to screen 0
OPCODE(0x5001): print R1 to screen 1
OPCODE(0x5002): assign the first character in the current keyboard buffer to R1
OPCODE(0x5011): replace R1 by underscore if it’s non-alphanumeric and then print R1 to screen 0

COMPLETE HERA CODE:

```
DLABEL(str2)
LP_STRING("hello")

CBON()

SET(R1, 0x0068) // R1 = h
OPCODE(0x5000) // print to TTY 0 using an opcode
```

```
SET(R1, 0x0069) // R1 = i
OPCODE(0x5000) // print to TTY 0 using an opcode

SET(R2, str2) // R2 = address of the string to print
LOAD(R3, 0, R2) // R3 = length of the string
INC(R2, 1) // R2 = Now the address of the initial char of the string

LABEL(loop)
LOAD(R1, 0, R2) // R1 = curr char of string.
OPCODE(0x5001) // print "hello" to tty using opcode
INC(R2, 1) // Go to next char
DEC(R3, 1) // Keeping track of # of chars to process
BNZ(loop)

SET(R7, 20) // R7 now stores the number of iterations left (wait time);
value is arbitrary

LABEL(COMPUTATION)
DEC(R7, 1)
BNZ(COMPUTATION) // continue loop
LABEL(NEXT1)

SETLO(R7, 0) // current keyboard = 0
SET(R4, 0x0a) // Storing newline character
SET(R5, 0x0d) // Storing return character
SET(R6, 0) // Whether or not to take a character from the keyboard.

// do something so that if OPCODE = 5002, and R1 would store the first
char in current buffer
// if opcode 5002 is present, then in the reg bank, a multiplexer can
take the current character from the buffer and store it in the
register.
// in reg bank: r1 = (OPCODE == 5002) V r1; write1 = write1 if OPCODE
== 5002, else buffer[0]

LABEL(PRINT_KEYBOARD0)
OPCODE(0x5002)
ADD(R0, R1, R0) // check if buffer is empty in case there is no newline
or return
BZ(EXIT) // if R1 empty, exit loop
SUB(R0, R4, R1) // Subtract the current character from r4 (newline)
BZ(EXIT) // If it's 0, exit. If not, continue.
SUB(R0, R5, R1) // Also test for R5 (carriage return)
BZ(EXIT)
OPCODE(0x5000) // otherwise, print R1 to TTY 0
```

```
SETLO(R6, 1) // consume the first character from keyboard 0
SETLO(R6, 0) // reset R6 to 0. Prevent first character of string from
being consumed by clock tick, before it has been put into register/RAM
BR(PRINT_KEYBOARD0)

LABEL(EXIT)
SETLO(R6, 1) // consume the special character (newline or return)
SETLO(R6, 0)

// similar process for above except we now have to write to RAM
SETLO(R7, 1) // current keyboard = 1
SETLO(R8, 1) // R8: address of current char from keyboard 1 in RAM.

LABEL(STORE_KEYBOARD1)
OPCODE(0x5002) // Store character in reg from keyboard 1 to R1
// Once stored, check for empty, newline, or return, then store in RAM
if not those cases.
ADD(R0, R1, R0) // check if buffer is empty in case there is no newline
or return
BZ(EXIT1) // if R1 empty, exit loop
SUB(R0, R4, R1) // Subtract the current character from r4 (newline)
BZ(EXIT1) // If it's 0, exit. If not, continue.
SUB(R0, R5, R1) // Also test for R5 (carriage return)
BZ(EXIT1)
STORE(R1, 0, R8) // store current char from R1 to RAM[R8]
INC(R8, 1) // increment R8 by 1
SETLO(R6, 2) // consume first char from keyboard 1
SETLO(R6, 0) // reset R6 to 0
BR(STORE_KEYBOARD1)

LABEL(EXIT1)
SETLO(R6, 2) // consume the special char from keyboard 1
SETLO(R6, 0) // reset R6 to 0
SETLO(R8, 1) // reset R8 to the address of the first char from keyboard
1

LABEL(PRINT_KEYBOARD1_TO_SCREEN1)
LOAD(R1, 0, R8) // load char from RAM to R1
ADD(R0, R1, R0)
BZ(EXIT2) // if R1 empty, exit loop

//todo: check alphanumeric in general
// check if R1 is a lowercase letter (i.e. if 0x60 < R1 < 0x7b). If
yes, convert R1 to uppercase by subtracting 0x20 from R1. Otherwise, go
ahead and print R1
```

```
SETLO(R2, 0x60)
SUB(R0, R1, R2)
BULE(SKIP_TO_PRINT)
SETLO(R2, 0x7b)
SUB(R0, R2, R1)
BULE(SKIP_TO_PRINT)
DEC(R1, 0x20)
LABEL(SKIP_TO_PRINT)
OPCODE(0x5001) // print R1 to tty 1
INC(R8, 1) // increment R8 by 1
BR(PRINT_KEYBOARD1_TO_SCREEN1)
LABEL(EXIT2)
SETLO(R8, 1) // reset R8 to the address of the first char from keyboard
1
LABEL(PRINT_KEYBOARD1_TO_SCREEN0)
LOAD(R1, 0, R8) // load char from RAM to R8
ADD(R1, R0, R1) // check if R1 is empty
BZ(EXIT3) // if empty, exit print loop
OPCODE(0x5011) // in hardware: OPCODE = 5011, R1 = underscore if R1 is
not alphanumeric
INC(R8, 1)
BR(PRINT_KEYBOARD1_TO_SCREEN0)
LABEL(EXIT3)
SETLO(R7, 1) // R7 = current keyboard
SETLO(R8, 1)

LABEL(CHECK_LOOP)
SUB(R7, R8, R7) // switch keyboard
OPCODE(0x5002) // set R1 as the first char from current keyboard
// if R1 is empty, skip to the next iteration of the loop
ADD(R0, R1, R0)
BZ(CHECK_LOOP)
// otherwise, if current keyboard = 0, convert R1 to upper-case if
needed and print to both terminals. If current keyboard = 1, convert R1
to lower-case if needed and print to both terminals
// check current keyboard (R7), if 0 continue and then loop, if 1 skip
to KEYBOARD_1 and then loop.
SUB(R0, R8, R7)
BZ(KEYBOARD_1)
// check if R1 is a lowercase letter (i.e. if 0x60 < R1 < 0x7b). If
yes, convert R1 to uppercase by subtracting 0x20 from R1. Otherwise, go
ahead and print R1
SETLO(R2, 0x60)
SUB(R0, R1, R2)
BULE(SKIP_TO_PRINT_FROM_KB0)
```

```
SETLO(R2, 0x7b)
SUB(R0, R2, R1)
BULE(SKIP_TO_PRINT_FROM_KB0)
DEC(R1, 0x20)

LABEL(SKIP_TO_PRINT_FROM_KB0)
OPCODE(0x5000) // print to terminal 0
OPCODE(0x5001) // print to terminal 1
// consume the first char from keyboard 0
SETLO(R6, 1)
SETLO(R6, 0)
BR(CHECK_LOOP)

LABEL(KEYBOARD_1)
// check if R1 is an uppercase letter (i.e. if 0x40 < R1 < 0x5b). If
yes, convert R1 to lowercase by adding 0x20 to R1. Otherwise, go ahead
and print R1
SETLO(R2, 0x40)
SUB(R0, R1, R2)
BULE(SKIP_TO_PRINT_FROM_KB1)
SETLO(R2, 0x5b)
SUB(R0, R2, R1)
BULE(SKIP_TO_PRINT_FROM_KB1)
INC(R1, 0x20)

LABEL(SKIP_TO_PRINT_FROM_KB1)
OPCODE(0x5000) // print to terminal 0
OPCODE(0x5001) // print to terminal 1
// consume the first char from keyboard 1
SETLO(R6, 2)
SETLO(R6, 0)
BR(CHECK_LOOP)
```

HERA code with detailed comments (same as above):

- The program first prints greetings to both terminals, with "`hi`" going to terminal 0 and something else ("`hello`", or a brief greeting in some other language) going to terminal 1.
- The `h` and the `i` should be placed in a register with `SET` (or `SETLO`); the characters from the second greeting should be extracted from a string defined with a `DLABEL/LP_STRING` definition in the data segment. When you're developing/debugging output, you may want to have a `HALT( )` right after these two steps, so you can get output working before attempting input.

```
DLABEL(str2)
LP_STRING("hello")

CBON()

SET(R1, 0x0068) // R1 = h
OPCODE(0x5000) // print to TTY 0 using an opcode
SET(R1, 0x0069) // R1 = i
OPCODE(0x5000) // print to TTY 0 using an opcode

SET(R2, str2) // R2 = address of the string to print
LOAD(R3, 0, R2) // R3 = length of the string
INC(R2, 1) // R2 = Now the address of the initial char of the string
LABEL(loop)
LOAD(R1, 0, R2) // R1 = curr char of string.
OPCODE(0x5001) // print "hello" to tty using opcode
INC(R2, 1) // Go to next char
DEC(R3, 1) // Keeping track of # of chars to process
BNZ(loop)
```

- The program then does some computation that takes a few seconds. Possibly the user types something at this point.

```
SET(R7, 20) // R7 now stores the number of iterations left (wait time);
value is arbitrary

LABEL(COMPUTATION)
DEC(R7, 1)
BNZ(COMPUTATION) // continue loop
LABEL(NEXT1)
```

- The program then reads characters from keyboard 0 and prints them immediately to screen 0, until it gets a newline or return.

```
SETLO(R7, 0) // current keyboard = 0
SET(R4, 0x0a) // Storing newline character
SET(R5, 0x0d) // Storing return character
SET(R6, 0) // Whether or not to take a character from the keyboard.

// do something so that if OPCODE = 5002, and R1 would store the first
char in current buffer
// if opcode 5002 is present, then in the reg bank, a multiplexer can
take the current character from the buffer and store it in the
register.
// in reg bank: r1 = (OPCODE == 5002) V r1; write1 = write1 if OPCODE
== 5002, else buffer[0]

LABEL(PRINT_KEYBOARD0)
OPCODE(0x5002)

ADD(R0, R1, R0) // check if buffer is empty in case there is no newline
or return
BZ(EXIT) // if R1 empty, exit loop
SUB(R0, R4, R1) // Subtract the current character from r4 (newline)
BZ(EXIT) // If its 0, exit. If not, continue.
SUB(R0, R5, R1) // Also test for R5 (carriage return)
BZ(EXIT)

OPCODE(0x5000) // otherwise, print R1 to TTY 0
SETLO(R6, 1) // consume the first character from keyboard 0
SETLO(R6, 0) // reset R6 to 0. Prevent first character of string from
being consumed by clock tick, before it has been put into register/RAM

BR(PRINT_KEYBOARD0)


LABEL(EXIT)
SETLO(R6, 1) // consume the special character (newline or return)
SETLO(R6, 0)
```

- The program then reads characters from keyboard 1, without immediately showing them, until the input character is a newline or linefeed. After reading the newline/linefeed, it prints the entire line to screen 1, but with all lower-case letters converted to upper-case, and prints the original line, with all characters that aren't numerals or letters replaced by underscores, to screen 0.

```
// similar process for above except we now have to write to RAM

SETLO(R7, 1) // current keyboard = 1
SETLO(R8, 1) // R8: address of current char from keyboard 1 in RAM.
```

```
LABEL(STORE_KEYBOARD1)
OPCODE(0x5002) // Store character in reg from keyboard 1 to R1

// Once stored, check for empty, newline, or return, then store in RAM
if not those cases.

ADD(R0, R1, R0) // check if buffer is empty in case there is no newline
or return
BZ(EXIT1) // if R1 empty, exit loop
SUB(R0, R4, R1) // Subtract the current character from r4 (newline)
BZ(EXIT1) // If its 0, exit. If not, continue.
SUB(R0, R5, R1) // Also test for R5 (carriage return)
BZ(EXIT1)

STORE(R1, 0, R8) // store current char from R1 to RAM[R8]
INC(R8, 1) // increment R8 by 1
SETLO(R6, 2) // consume first char from keyboard 1
SETLO(R6, 0) // reset R6 to 0
BR(STORE_KEYBOARD1)

LABEL(EXIT1)
SETLO(R6, 2) // consume the special char from keyboard 1
SETLO(R6, 0) // reset R6 to 0
SETLO(R8, 1) // reset R8 to the address of the first char from keyboard
1

LABEL(PRINT_KEYBOARD1_TO_SCREEN1)
LOAD(R1, 0, R8) // load char from RAM to R1
ADD(R0, R1, R0)
BZ(EXIT2) // if R1 empty, exit loop

//todo: check alphanumeric in general

// check if R1 is a lowercase letter (i.e. if 0x60 < R1 < 0x7b). If
yes, convert R1 to uppercase by subtracting 0x20 from R1. Otherwise, go
ahead and print R1
SETLO(R2, 0x60)
SUB(R0, R1, R2)
BULE(SKIP_TO_PRINT)
SETLO(R2, 0x7b)
SUB(R0, R2, R1)
BULE(SKIP_TO_PRINT)
DEC(R1, 0x20)

LABEL(SKIP_TO_PRINT)
OPCODE(0x5001) // print R1 to tty 1
INC(R8, 1) // increment R8 by 1
BR(PRINT_KEYBOARD1_TO_SCREEN1)

LABEL(EXIT2)
```

```
SETLO(R8, 1) // reset R8 to the address of the first char from keyboard
1

LABEL(PRINT_KEYBOARD1_TO_SCREEN0)

LOAD(R1, 0, R8) // load char from RAM to R8
ADD(R1, R0, R1) // check if R1 is empty
BZ(EXIT3) // if empty, exit print loop
OPCODE(0x5011) // in hardware: OPCODE = 5011, R1 = underscore if R1 is
not alphanumeric
INC(R8, 1)
BR(PRINT_KEYBOARD1_TO_SCREEN0)

LABEL(EXIT3)

SETLO(R7, 1) // R7 = current keyboard
SETLO(R8, 1)
```

- The program then enters a loop in which it checks each keyboard to see if it has a character, and if so, prints that character to both screens, but with all letters from keyboard 0 showing up in upper-case on both terminals, and all letters from keyboard 1 showing up in lower-case on both terminals.

```
LABEL(CHECK_LOOP)

SUB(R7, R8, R7) // switch keyboard
OPCODE(0x5002) // set R1 as the first char from current keyboard

// if R1 is empty, skip to the next iteration of the loop
ADD(R0, R1, R0)
BZ(CHECK_LOOP)

// otherwise, if current keyboard = 0, convert R1 to upper-case if
needed and print to both terminals. If current keyboard = 1, convert R1
to lower-case if needed and print to both terminals

// check current keyboard (R7), if 0 continue and then loop, if 1 skip
to KEYBOARD_1 and then loop.

SUB(R0, R8, R7)
BZ(KEYBOARD_1)

// check if R1 is a lowercase letter (i.e. if 0x60 < R1 < 0x7b). If
yes, convert R1 to uppercase by subtracting 0x20 from R1. Otherwise, go
ahead and print R1
SETLO(R2, 0x60)
SUB(R0, R1, R2)
BULE(SKIP_TO_PRINT_FROM_KB0)
```

```
SETLO(R2, 0x7b)
SUB(R0, R2, R1)
BULE(SKIP_TO_PRINT_FROM_KB0)
DEC(R1, 0x20)
LABEL(SKIP_TO_PRINT_FROM_KB0)

OPCODE(0x5000) // print to terminal 0
OPCODE(0x5001) // print to terminal 1

// consume the first char from keyboard 0

SETLO(R6, 1)
SETLO(R6, 0)
BR(CHECK_LOOP)

LABEL(KEYBOARD_1)

// check if R1 is an uppercase letter (i.e. if 0x40 < R1 < 0x5b). If
yes, convert R1 to lowercase by adding 0x20 to R1. Otherwise, go ahead
and print R1

SETLO(R2, 0x40)
SUB(R0, R1, R2)
BULE(SKIP_TO_PRINT_FROM_KB1)
SETLO(R2, 0x5b)
SUB(R0, R2, R1)
BULE(SKIP_TO_PRINT_FROM_KB1)
INC(R1, 0x20)

LABEL(SKIP_TO_PRINT_FROM_KB1)

OPCODE(0x5000) // print to terminal 0
OPCODE(0x5001) // print to terminal 1

// consume the first char from keyboard 1
SETLO(R6, 2)
SETLO(R6, 0)
BR(CHECK_LOOP)
```

INSTRUCTIONS (IN ROM):

Put into ROM:

```
3160
e168
f100
5000
e169
f100
5000
e201
```

```
f2c0
4302
3280
4102
5001
3280
33c0
eb0b
fb00
190b
e714
f700
37c0
eb14
fb00
190b
e700
e40a
f400
e50d
f500
e600
f600
5002
a010
eb32
fb00
180b
b041
eb32
fb00
180b
b051
eb32
fb00
180b
5000
e601
e600
eb1f
fb00
100b
e601
e600
e701
```

```
e801
5002
a010
eb4a
fb00
180b
b041
eb4a
fb00
180b
b051
eb4a
fb00
180b
6108
3880
e602
e600
eb36
fb00
100b
e602
e600
e801
4108
a010
eb62
fb00
180b
e260
b012
eb5d
fb00
160b
e27b
b021
eb5d
fb00
160b
31df
5001
3880
eb4d
fb00
100b
```

e801
4108
a101
eb6d
fb00
180b
5011
3880
eb63
fb00
100b
e701
e801
b787
5002
a010
eb6f
fb00
180b
b087
eb8b
fb00
180b
e260
b012
eb84
fb00
160b
e27b
b021
eb84
fb00
160b
31df
5000
5001
e601
e600
eb6f
fb00
100b
e240
b012
eb96
fb00

160b
e25b
b021
eb96
fb00
160b
319f
5000
5001
e602
e600
eb6f
fb00
100b