Name: Cameron Peterson-Zopf

PROGRAMMING ASSIGNMENT 3

Discussion on a high level with your colleagues is encouraged. Make sure the work submitted is your own. When in doubt, ask a TA or the instructor. If you are not sure what constitutes academic dishonesty, please refer to the AISC web site: https://aisc.uci.edu/.

You can fill out your answers below in text, paste screenshots, and/or include images (make sure the image is right side up & legible).

This homework covers:

- LC-3 Machine Language to Assembly – Part 1

AISC

Please initial here to indicate you understand UCI's Academic Integrity Policy and confirm that this is your own work you are submitting (this counts for points): CPZ

UPDATED CODE SCREENSHOT

Final assembly code w/ comments (screenshot or pasted in)

hello-buddy-login.asm

```
;Problem 9.30
;
; Program asks user to enter their name, then greets the user:
; > Please enter your name: <user input>
; > Hello, <user input>
; Encription: encripted by having the ASCII value of the character be stored as #10 more.
;
; R0: holds the input (from TRAP x20) & uses as output (for TRAP x21) and starting addresses for output strings
; R1: initializes to the starting of HELLO string & ends up pointing to the end of the string. Will be saved in memory.
; R2: temporary register used for intermediate calculations. Also used as a counter for attempts. Will be saved in memory.
; R3: temporary register used for intermediate calculations. Also used to store info to jump to SUCCESS. Will be saved in memory.
; R4: temporary register used for intermediate calculations. Also used to store info to jump to YESPW
; R5: temporary register used for intermediate calculations
; R6: temporary register used for intermediate calculations
; R7: reserved for subroutines

        .ORIG   x3000
        LEA     R1, HELLO       ; starting address of string HELLO (going to find end of string for appending)
        AND     R2, R2, x0000   ; R2<-0
        ADD     R2, R2, #3      ; R2<-3
        ST      R2, Save        ; save R2 as it will be used in subroutines for user
        ST      R2, SaveP       ; save R2 as it will be used in subroutines for password
        LEA     R3, SUCCESS     ; save where SUCCESS is
        ST      R3, Save1       ; save R3 as it will be used in subroutines
        LEA     R4, YESPW       ; save where YESPW is
        ST      R4, Save2       ; save R4 as it will be used in subroutines
```

```
; get to end of HELLO string for writing/appending input
AGAIN   LDR     R2,R1,#0        ; load char at hello address into R2
        BRz     NEXT            ; if done w/ string (x0000...NULL), go to NEXT
        ADD     R1,R1,#1        ; increment hello address..so you can store next character into R2, overwriting previous
        ST      R1, SaveR1      ; save value of R1 to point to starting location for user input.
        BR      AGAIN
        ; when we branch to NEXT, R1 will continue the address of the NULL character in the hello string.

; print prompt for username input
NEXT    LEA     R0,PROMPT       ; get starting address of prompt for user
        TRAP    x22             ; PUTS (output entire welcome message)


;Get user input and compare
        JSR     FuUS            ; call reading in of username
        JSR     FuCom1          ; compare against USER1
        JSR     FuCom2          ; compare against USER2
        JSR     FuCom3          ; compare against USER3
; If not a valid username...
        LD      R2, Save        ; load back R2.
        ADD     R2, R2, #-1     ; decrement R2
        BRz     FINISH          ; tried 3 times
        ST      R2, Save        ; save R2 again
        LEA     R0,FAIL         ; get starting address of fail message for user

        TRAP    x22             ; PUTS (output)
        JSR     ClrU            ; clear previous username
        BR      NEXT            ; back to start
FINISH  LEA     R0,FAIL3        ; get starting address of fail 3 times message for user
        TRAP    x22             ; PUTS (output)
        BR      BYE
; Get password input and compare
SUCCESS LEA     R0, PWPROMPT
        TRAP    x22
        JSR     FuPW            ; call reading in of password
        JSR     PWCom1          ; compare against PW1
        JSR     PWCom2          ; compare against PW2
        JSR     PWCom3          ; compare against PW3
; If not a valid password...
        LD      R2, SaveP       ; load back R2.
        ADD     R2, R2, #-1     ; decrement R2
        BRz     FINISH1         ; tried 3 times
        ST      R2, SaveP       ; save R2 again
        LEA     R0,FAIL         ; get starting address of fail message for user
        TRAP    x22             ; PUTS (output)
        JSR     ClrP            ; clear previous password
        BR      SUCCESS         ; back to start
FINISH1 LEA     R0,FAIL3        ; get starting address of fail 3 times message for user
        TRAP    x22             ; PUTS (output)
        BR      BYE

; password was correct
YESPW   LEA     R0, HELLO       ; address of HELLO Prompt
        TRAP    x22             ; PUTS...since we overwrote the 1st NULL, this will print out both Hello, name
        LEA     R0, LOGIN       ; address of LOGIN Prompt
        TRAP    x22
BYE     TRAP    x25             ; HALT (stop program)


;************************Subroutines For Writing Username and PW*************************************
;function for writing username
FuUS    LD      R3, NEGENTER    ; store (through ld) NEGENTER into R3 for termination comparing later
        LD      R1, SaveR1      ; load in starting address for input.
AGAIN2  TRAP    x20             ; GETC (gather user input one char at a time)
        TRAP    x21             ; OUT (output char for user on console)
        ADD     R2,R0,R3        ; check if user pressed ENTER key. If R2 = 0 => R0 = x0A
        BRz     ADDNULL         ; if they did, we are done
        STR     R0,R1,#0        ; store value in R0 into memory. R1 is first pointing to NULL character, which will be overwritten.
        ADD     R1,R1,#1        ; increment address of R1 so that we can write to the next available spot (in .BLKW #25 area)
        BR      AGAIN2
ADDNULL AND     R6, R6, x0000   ; R6<-0
        STR     R6, R1, #0      ; end with NULL
        RET

```

```
;function for writing password
FuPW    LD      R3, NEGENTER    ; store (through ld) NEGENTER into R3 for termination comparing later
        LEA     R5, SPACE       ; store in empty space. Load back in starting location.
AGAIN3  TRAP    x20             ; GETC (gather user input one char at a time)
        ADD     R2,R0,R3        ; check if user pressed ENTER key. If R2 = 0 => R0 = x0A
        BRz     NULLPW          ; if they did, we are done
        STR     R0,R5,#0        ; store value in R0 (user input) into memory.
        ADD     R5,R5,#1        ; increment address of R5 so that we can write to the next available spot
        BR      AGAIN3
NULLPW  AND     R6, R6, x0000   ; R6<-0
        STR     R6, R5, #0      ; end with NULL
        RET

;*****************Subroutines for Clearing Wrong Username and PW*****************
;function for clearing username
ClrU    LD      R3, NEGENTER    ; store (through ld) NEGENTER into R3 for termination comparing later
        LD      R1, SaveR1      ; load in starting address for input.
        AND     R2,R2, x0000    ; R2<-0
        ADD     R4,R2, #15      ; R4<-15
        ADD     R4, R4, #10     ; R4<-25
AGAIN4  STR     R2,R1,#0        ; Set value to zero.
        ADD     R1,R1,#1        ; increment address of R1 so that we can write to the next available spot (in .BLKW #25 area)
        ADD     R4, R4, #-1     ;decrement R4 counter
        BRz     #1              ; done clearing all 25 spaces.
        BR      AGAIN4
        RET


;function for clearing password
ClrP    LD      R3, NEGENTER    ; store (through ld) NEGENTER into R3 for termination comparing later
        LEA      R5, SPACE       ; load in starting address for input pw
        AND     R2,R2, x0000    ; R2<-0
        ADD     R4,R2, #15      ; R4<-15
        ADD     R4,R4, #10      ; R4<-25
AGAIN5  STR     R2,R5,#0        ; Set value to zero.
        ADD     R5,R5,#1        ; increment address of R1 so that we can write to the next available spot (in .BLKW #25 area)
        ADD     R4, R4, #-1     ;decrement R4 counter
        BRz     #1              ; done clearing all 25 spaces.
        BR      AGAIN5
        RET

;************************Subroutines For Checking Username*************************************
; function for comparing input to USER1
FuCom1  LEA     R2, USER1       ; load starting address of USER1 into R2
        LEA     R3, INPUT       ; load address of 2nd character of input (1st character overwrote NULL)
        ADD     R3, R3, #-1     ; Now we have starting address of input
Loop1   LDR     R4, R2, #0      ; R4<-M[R2] // R4<-M[address]...store character from USER1 into R4
        ADD     R2, R2, #1      ; increments R2 to next address
        LDR     R5, R3, #0      ; R5<-M[R3]...store character from input into R5
        ADD     R3, R3, #1      ; increments R3 to next address
        NOT     R5, R5
        ADD     R5, R5, #1      ; 2s comp => negative
        ADD     R6, R4, R5      ; If R6 = 0 => same character

        BRnp    NOTUSER1        ; not a valid username
        ADD     R4, R4, #0      ; List R4 again for branch statement
        BRz     SUCCESS         ; if R4 and R5 is the NULL character, input is done, and matched with USER1
        BR      Loop1
NOTUSER1 RET                     ; the input was not this username, try to compare against the next one.

; function for comparing input to USER2
FuCom2  LEA     R2, USER2       ; load starting address of USER2 into R2
        LEA     R3, INPUT       ; load address of 2nd character of input (1st character overwrote NULL)
        ADD     R3, R3, #-1     ; Now we have starting address of input
Loop2   LDR     R4, R2, #0      ; R4<-M[R2] // R4<-M[address]...store character from USER2 into R4
        ADD     R2, R2, #1      ; increments R2 to next address
        LDR     R5, R3, #0      ; R5<-M[R3]...store character from input into R5
        ADD     R3, R3, #1      ; increments R3 to next address
        NOT     R5, R5
        ADD     R5, R5, #1      ; 2s comp => negative
        ADD     R6, R4, R5      ; If R6 = 0 => same character
        BRnp    NOTUSER2        ; not a valid username
        ADD     R4, R4, #0      ; List R4 again for branch statement
        BRz     SUCJMP1         ; if R5 is the NULL character, input is done, and matched with USER2
        BR      Loop2
NOTUSER2 RET                     ; the input was not this username, try to compare against the next one.
SUCJMP1 LD      R3, Save1       ; load back R3, which is address for SUCCESS.
        JMP     R3              ; jump to where success is.

```

```
178 ; function for comparing input to USER3
179 FuCom3  LEA     R2, USER3       ; load starting address of USER3 into R2
180         LEA     R3, INPUT       ; load address of 2nd character of input (1st character overwrote NULL)
181         ADD     R3, R3, #-1     ; Now we have starting address of input
182 Loop3   LDR     R4, R2, #0      ; R4<-M[R2] // R4<-M[address]...store character from USER3 into R4
183         ADD     R2, R2, #1      ; increments R2 to next address
184         LDR     R5, R3, #0      ; R5<-M[R3]...store character from input into R5
185         ADD     R3, R3, #1      ; increments R3 to next address
186         NOT     R5, R5
187         ADD     R5, R5, #1      ; 2s comp => negative
188         ADD     R6, R4, R5      ; If R6 = 0 => same character
189         BRnp    NOTUSER3         ; not a valid username
190         ADD     R4, R4, #0      ; List R4 again for branch statement
191         BRz     SUCCJMP          ; if R5 is the NULL character, input is done, and matched with USER3
192         BR      Loop3
193 NOTUSER3 RET                     ; the input was not this username, try to compare against the next one.
194 SUCCJMP LD      R3, Save1        ; load back R3, which is address for SUCCESS.
195         JMP     R3               ; jump to where success is.
196
```

```
197 ;***************memory storage*****************************************
198 NEGENTER    .FILL       xFFF6       ; -x0A...negative of ENTER key
199 Save        .BLKW       1
200 SaveP       .BLKW       1
201 Save1       .BLKW       1
202 Save2       .BLKW       1
203 SaveR1      .BLKW       1
204 SPACE       .BLKW       #25
205 HELLO       .STRINGZ    "Hello, "
206 INPUT       .BLKW       #25
207 PROMPT      .STRINGZ    "Enter your username: "
208 FAIL        .STRINGZ    "Not valid, try again "
209 FAIL3       .STRINGZ    "3 attempts failed, goodbye"
210 PWPROMPT    .STRINGZ    "Enter your password: "
211 LOGIN       .STRINGZ    ". You have logged in"
212 USER1       .STRINGZ    "panteater"
213 PW1         .STRINGZ    "ujyjw"           ; "peter"
214 USER2       .STRINGZ    "qv"
215 PW2         .STRINGZ    "mjqqtymjwj&"     ; "hellothere!"
216 USER3       .STRINGZ    "john"
217 PW3         .STRINGZ    "xrnym"           ;  "smith"
218
219
```

```
220 ;************************Subroutines For Checking PW*************************************
221 ; function to jump to where SUCESSS is.
222 PWJMP   LD      R4, Save2       ; load back R4, which is address for YESPW.
223         JMP     R4              ; jump to where YESPW is.
224
225 ; function for comparing input to PW1
226 PWCom1  LEA     R2, PW1         ; load starting address of PW1 into R2
227         LEA     R3, SPACE       ; load address of input PW
228 Loop4   LDR     R4, R2, #0      ; R4<-M[R2] // R4<-M[address]...store character from PW1 into R4
229         BRz     #1              ; If R4 = 0 ==> NULL character, no conversion needed, skip.
230         ADD     R4, R4, #-5     ; subtract by 5 as the encription increased it by 5.
231         ADD     R2, R2, #1      ; increments R2 to next address
232         LDR     R5, R3, #0      ; R5<-M[R3]...store character from input into R5
233         ADD     R3, R3, #1      ; increments R3 to next address
234         NOT     R5, R5
235         ADD     R5, R5, #1      ; 2s comp => negative
236         ADD     R6, R4, R5      ; If R6 = 0 => same character
237         BRnp    NOTPW1          ; not a valid username
238         ADD     R4, R4, #0      ; List R4 again for branch statement
239         BRz     PWJMP           ; if R4 and R5 is the NULL character, input is done, and matched with PW1
240         BR      Loop4
241 NOTPW1  RET                      ; the input was not this username, try to compare against the next one.
242
243
```

```
244 ; function for comparing input to PW2
245 PWCom2  LEA     R2, PW2         ; load starting address of PW2 into R2
246         LEA     R3, SPACE       ; load address of input PW
247 Loop5   LDR     R4, R2, #0      ; R4<-M[R2] // R4<-M[address]...store character from PW2 into R4
248         BRz     #1              ; If R4 = 0 ==> NULL character, no conversion needed, skip.
249         ADD     R4, R4, #-5     ; subtract by 5 as the encription increased it by 5.
250         ADD     R2, R2, #1      ; increments R2 to next address
251         LDR     R5, R3, #0      ; R5<-M[R3]...store character from input into R5
252         ADD     R3, R3, #1      ; increments R3 to next address
253         NOT     R5, R5
254         ADD     R5, R5, #1      ; 2s comp => negative
255         ADD     R6, R4, R5      ; If R6 = 0 => same character
256         BRnp    NOTPW2          ; not a valid username
257         ADD     R4, R4, #0      ; List R4 again for branch statement
258         BRz     PWJMP           ; if R4 and R5 is the NULL character, input is done, and matched with PW2
259         BR      Loop5
260 NOTPW2  RET                     ; the input was not this username, try to compare against the next one.
261
```

```
262 ; function for comparing input to PW3
263 PWCom3  LEA     R2, PW3         ; load starting address of PW3 into R2
264         LEA     R3, SPACE       ; load address of input PW
265 Loop6   LDR     R4, R2, #0      ; R4<-M[R2] // R4<-M[address]...store character from PW3 into R4
266         BRz     #1              ; If R4 = 0 ==> NULL character, no conversion needed, skip.
267         ADD     R4, R4, #-5     ; subtract by 5 as the encription increased it by 5.
268         ADD     R2, R2, #1      ; increments R2 to next address
269         LDR     R5, R3, #0      ; R5<-M[R3]...store character from input into R5
270         ADD     R3, R3, #1      ; increments R3 to next address
271         NOT     R5, R5
272         ADD     R5, R5, #1      ; 2s comp => negative
273         ADD     R6, R4, R5      ; If R6 = 0 => same character
274         BRnp    NOTPW3          ; not a valid username
275         ADD     R4, R4, #0      ; List R4 again for branch statement
276         BRz     PWJMP           ; if R4 and R5 is the NULL character, input is done, and matched with PW3
277         BR      Loop6
278 NOTPW3  RET                     ; the input was not this username, try to compare against the next one.
279
280             .END
```

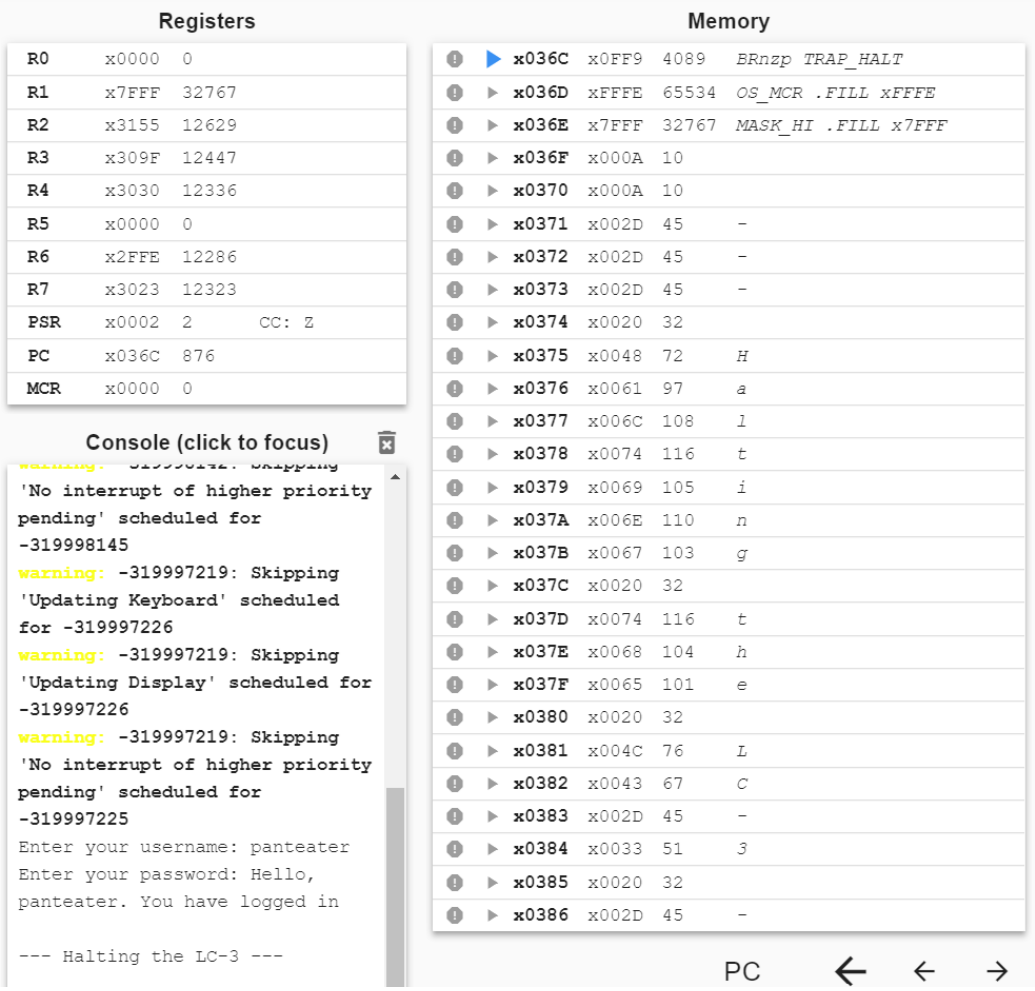
SCREENSHOT OF THE LOCATION OF THE USERNAMES & PASSWORDS (ENCRYPTED PASSWORDS)

The encrypted passwords are the ones in the code. The actual passwords are in the comments.

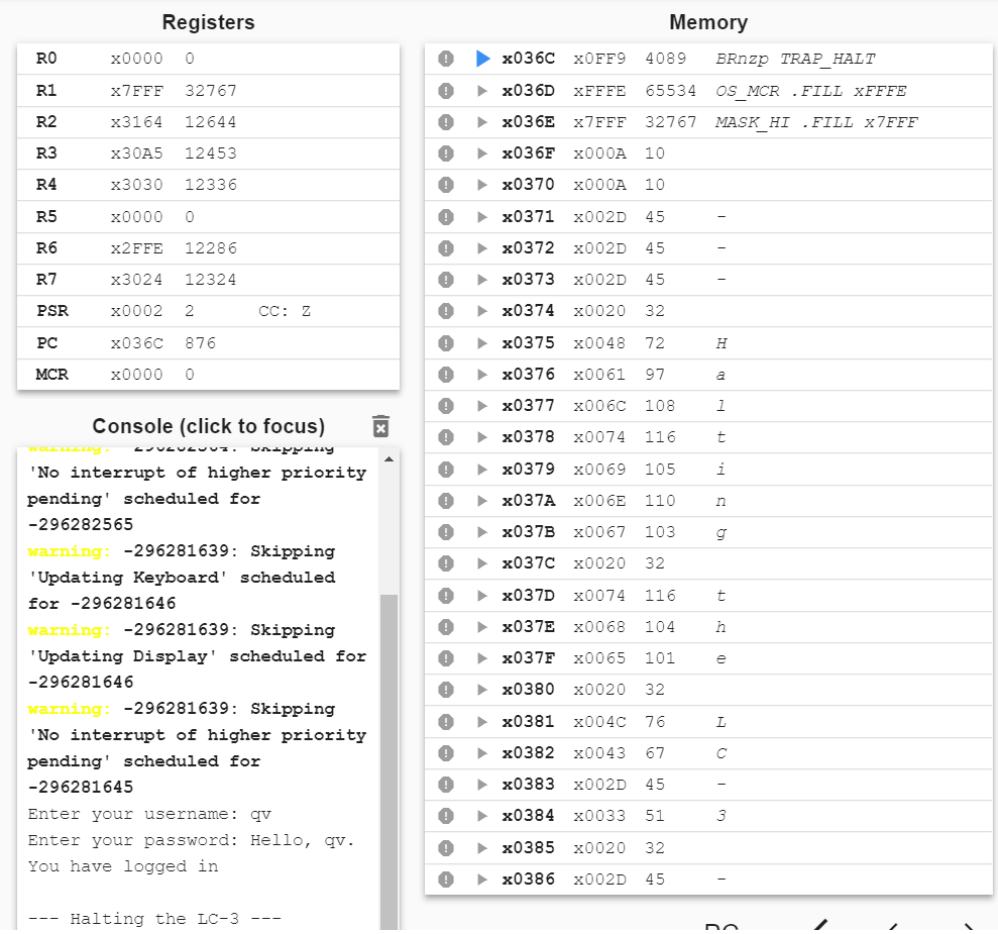
```
;***************memory storage*****************************************
NEGENTER    .FILL       xFFF6       ; -x0A...negative of ENTER key
Save        .BLKW       1
SaveP       .BLKW       1
Save1       .BLKW       1
Save2       .BLKW       1
SaveR1      .BLKW       1
SPACE       .BLKW       #25
HELLO       .STRINGZ    "Hello, "
INPUT       .BLKW       #25
PROMPT      .STRINGZ    "Enter your username: "
FAIL        .STRINGZ    "Not valid, try again "
FAIL3       .STRINGZ    "3 attempts failed, goodbye"
PWPROMPT    .STRINGZ    "Enter your password: "
LOGIN       .STRINGZ    ". You have logged in"
USER1       .STRINGZ    "panteater"
PW1         .STRINGZ    "ujyjw"           ; "peter"
USER2       .STRINGZ    "qv"
PW2         .STRINGZ    "mjqqtymjwj&"     ; "hellothere!"
USER3       .STRINGZ    "john"
PW3         .STRINGZ    "xrnym"           ;  "smith"

```

SCREENSHOT OF A USER LOGGING IN SUCCESSFULLY

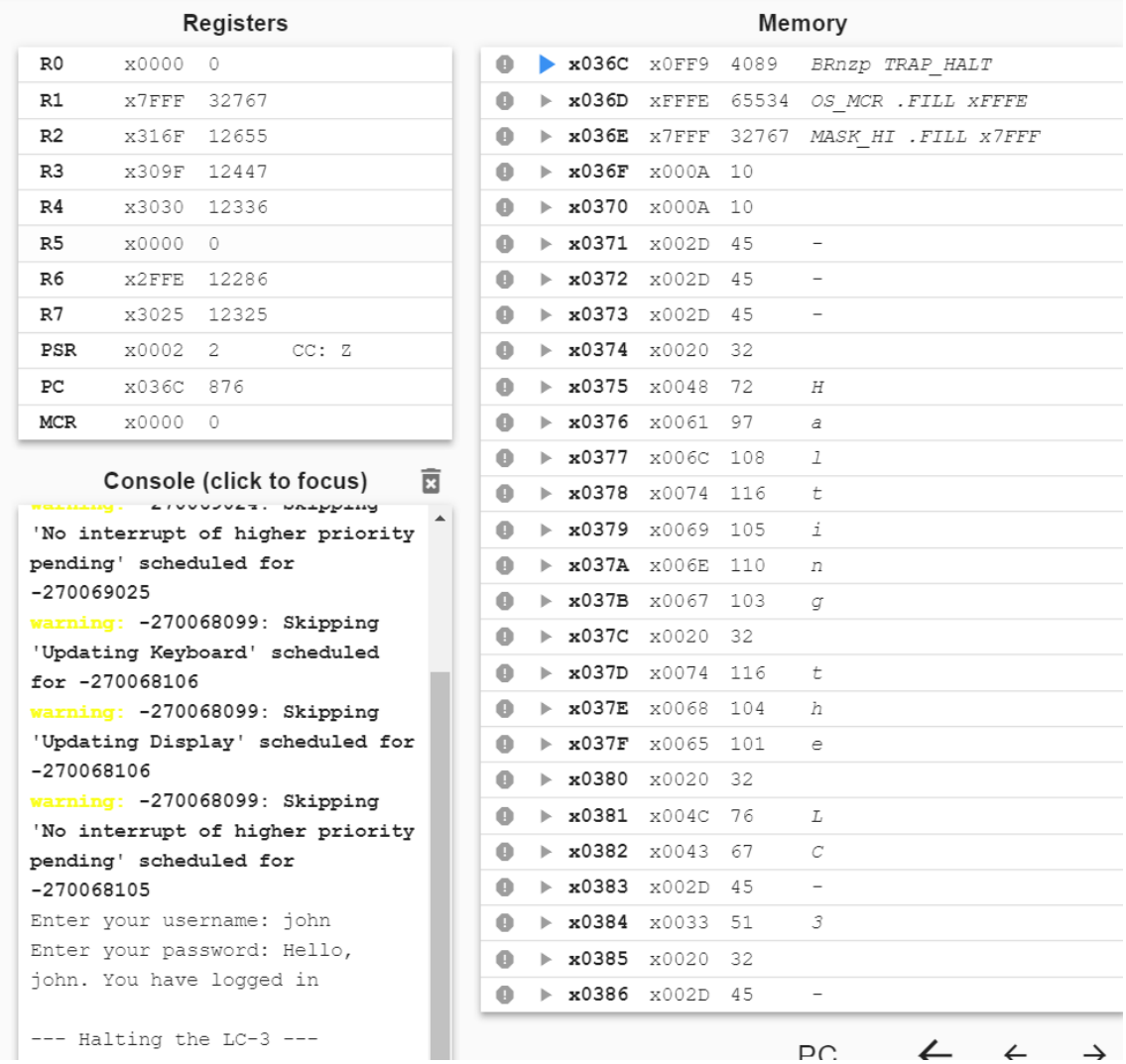
User panteater:


Registers
R0 x0000 0
R1 x7FFF 32767
R2 x3155 12629
R3 x309F 12447
R4 x3030 12336
R5 x0000 0
R6 x2FFE 12286
R7 x3023 12323
PSR x0002 2 CC: Z
PC x036C 876
MCR x0000 0
Console (click to focus)
'No interrupt of higher priority pending' scheduled for -319998145
warning: -319997219: Skipping 'Updating Keyboard' scheduled for -319997226
warning: -319997219: Skipping 'Updating Display' scheduled for -319997226
warning: -319997219: Skipping 'No interrupt of higher priority pending' scheduled for -319997225
Enter your username: panteater
Enter your password: Hello, panteater. You have logged in
--- Halting the LC-3 ---
Memory
x036C x0FF9 4089 BRnzp TRAP_HALT
x036D xFFFE 65534 OS_MCR .FILL xFFFE
x036E x7FFF 32767 MASK_HI .FILL x7FFF
x036F x000A 10
x0370 x000A 10
x0371 x002D 45 -
x0372 x002D 45 -
x0373 x002D 45 -
x0374 x0020 32
x0375 x0048 72 H
x0376 x0061 97 a
x0377 x006C 108 l
x0378 x0074 116 t
x0379 x0069 105 i
x037A x006E 110 n
x037B x0067 103 g
x037C x0020 32
x037D x0074 116 t
x037E x0068 104 h
x037F x0065 101 e
x0380 x0020 32
x0381 x004C 76 L
x0382 x0043 67 C
x0383 x002D 45 -
x0384 x0033 51 3
x0385 x0020 32
x0386 x002D 45 -
PC

User qv:

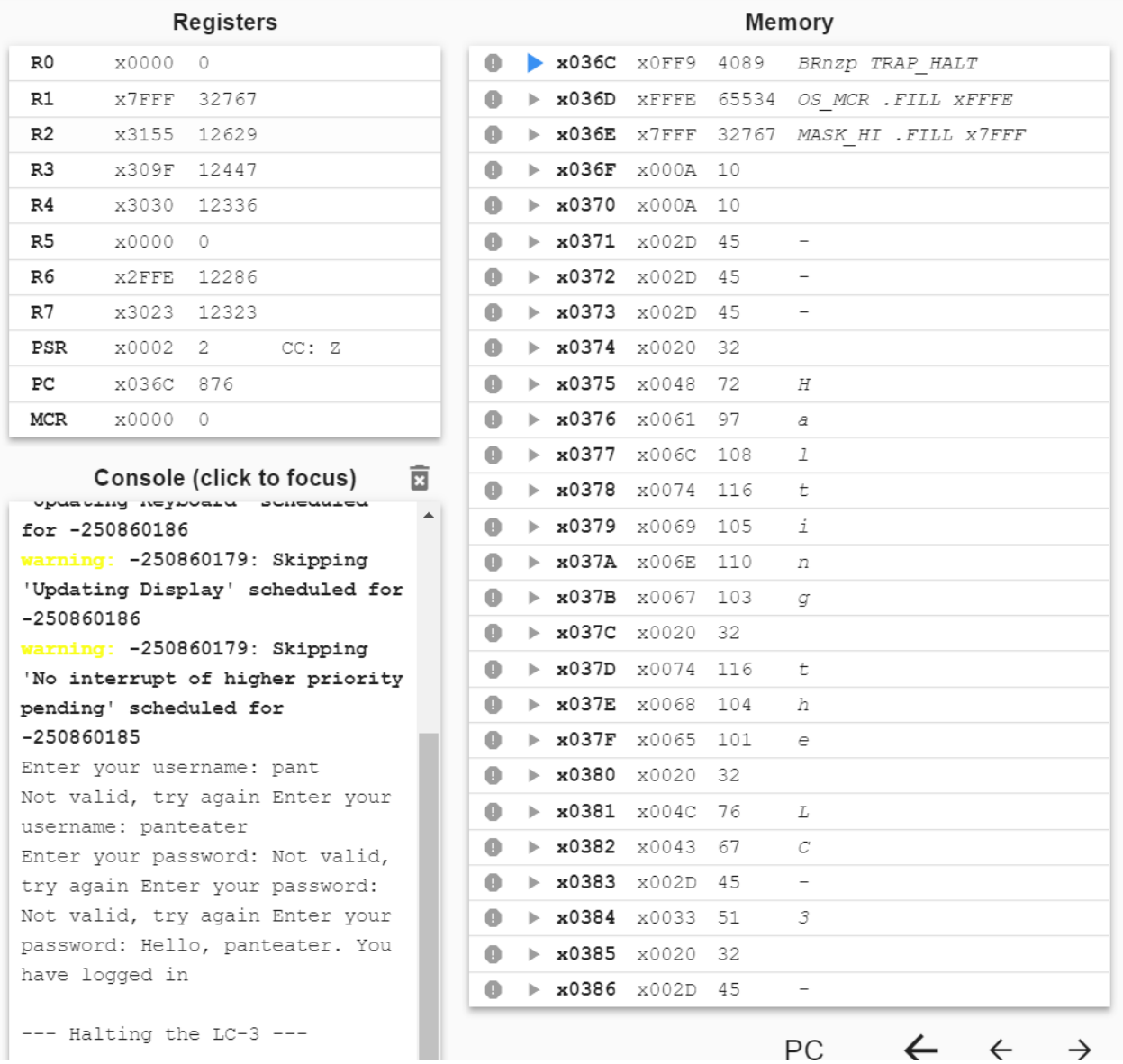

Registers
R0 x0000 0
R1 x7FFF 32767
R2 x3164 12644
R3 x30A5 12453
R4 x3030 12336
R5 x0000 0
R6 x2FFE 12286
R7 x3024 12324
PSR x0002 2 CC: Z
PC x036C 876
MCR x0000 0
Console (click to focus)
'No interrupt of higher priority pending' scheduled for -296282565
warning: -296281639: Skipping 'Updating Keyboard' scheduled for -296281646
warning: -296281639: Skipping 'Updating Display' scheduled for -296281646
warning: -296281639: Skipping 'No interrupt of higher priority pending' scheduled for -296281645
Enter your username: qv
Enter your password: Hello, qv.
You have logged in
--- Halting the LC-3 ---
Memory
x036C x0FF9 4089 BRnzp TRAP_HALT
x036D xFFFE 65534 OS_MCR .FILL xFFFE
x036E x7FFF 32767 MASK_HI .FILL x7FFF
x036F x000A 10
x0370 x000A 10
x0371 x002D 45 -
x0372 x002D 45 -
x0373 x002D 45 -
x0374 x0020 32
x0375 x0048 72 H
x0376 x0061 97 a
x0377 x006C 108 l
x0378 x0074 116 t
x0379 x0069 105 i
x037A x006E 110 n
x037B x0067 103 g
x037C x0020 32
x037D x0074 116 t
x037E x0068 104 h
x037F x0065 101 e
x0380 x0020 32
x0381 x004C 76 L
x0382 x0043 67 C
x0383 x002D 45 -
x0384 x0033 51 3
x0385 x0020 32
x0386 x002D 45 -

User john

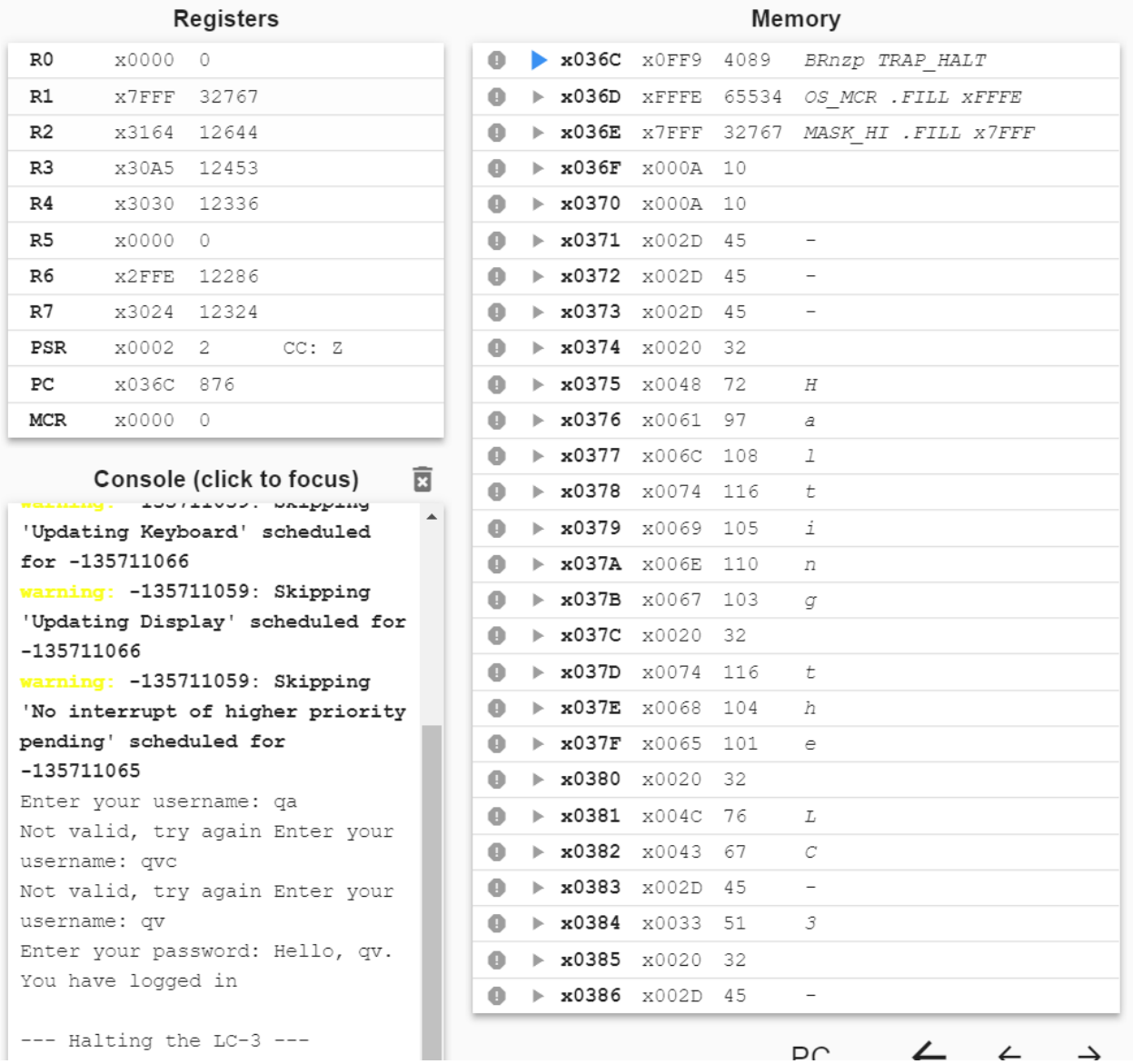


SCREENSHOT OF A USER LOGGING IN UNSUCCESSFULLY (EITHER WRONG USER NAME, WRONG PASSWORD, OR BOTH)

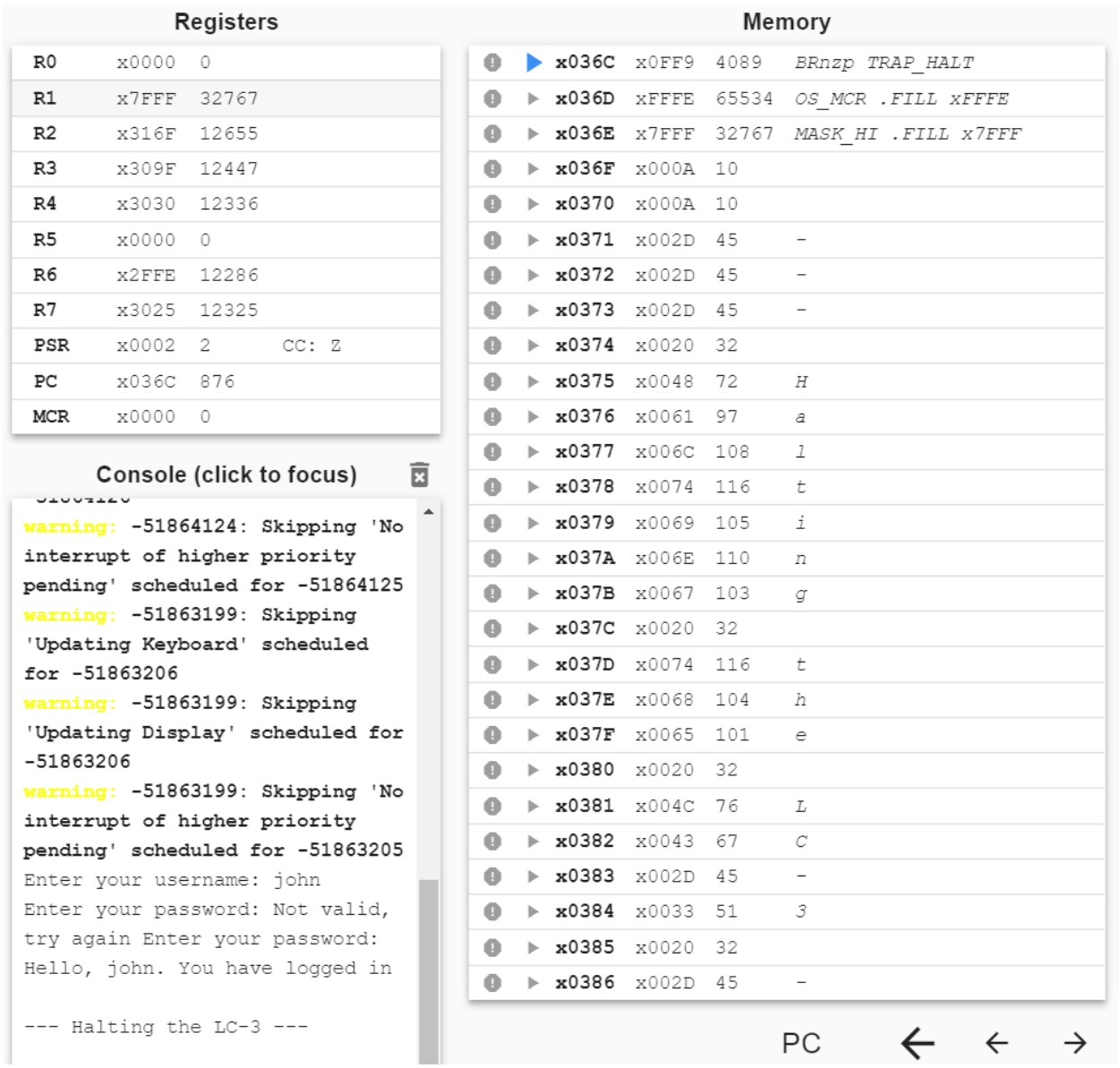
Testing with panteater, username is typed wrong once and password is typed wrong twice:



Testing with qv, username typed in wrong twice, password is right first time:



Testing with john, username right, password wrong once:


Registers
R0 x0000 0
R1 x7FFF 32767
R2 x316F 12655
R3 x309F 12447
R4 x3030 12336
R5 x0000 0
R6 x2FFE 12286
R7 x3025 12325
PSR x0002 2 CC: Z
PC x036C 876
MCR x0000 0
Console (click to focus)
warning: -51864124: Skipping 'No interrupt of higher priority pending' scheduled for -51864125
warning: -51863199: Skipping 'Updating Keyboard' scheduled for -51863206
warning: -51863199: Skipping 'Updating Display' scheduled for -51863206
warning: -51863199: Skipping 'No interrupt of higher priority pending' scheduled for -51863205
Enter your username: john
Enter your password: Not valid, try again Enter your password:
Hello, john. You have logged in
--- Halting the LC-3 ---
Memory
x036C x0FF9 4089 BRnzp TRAP_HALT
x036D xFFFE 65534 OS_MCR .FILL xFFFE
x036E x7FFF 32767 MASK_HI .FILL x7FFF
x036F x000A 10
x0370 x000A 10
x0371 x002D 45 -
x0372 x002D 45 -
x0373 x002D 45 -
x0374 x0020 32
x0375 x0048 72 H
x0376 x0061 97 a
x0377 x006C 108 l
x0378 x0074 116 t
x0379 x0069 105 i
x037A x006E 110 n
x037B x0067 103 g
x037C x0020 32
x037D x0074 116 t
x037E x0068 104 h
x037F x0065 101 e
x0380 x0020 32
x0381 x004C 76 L
x0382 x0043 67 C
x0383 x002D 45 -
x0384 x0033 51 3
x0385 x0020 32
x0386 x002D 45 -
PC