Project 4

Jacob Abel

December 11, 2017

Contents

W

r0

Source Code

```
1  //  Name: Jacob Abel
2  //  ID: 6453
3  //  Date: Monday, 2017-12-11
4  //
5  //  Starter source code program for ECE 2504
6  //  Project 4
7  //
8  //  The on-line assembler is experimental and minimal.
9  //  No guarantees are made about its correctness.
10 //
11 //
12 //  Do not change the following segment of code from here
13 //  to the comment "CHANGE HERE" below.
14 ldi r0, 3
15 shl r0, r0
16 shl r0, r0
17 inc r0, r0
18 shl r0, r0
19 jmp r0
20 // The following set of load instructions read the final values
      of the variables in
21 // memory locations into r1-r7 so that we can see them on the
      LEDs.
22 // Your code must jump to this point after it has stored the
      results in data memory.
23 // You should jump to location 6.
24 // address 0x06
25 xor r0, r0, r0
26 ld r1
```

W 4r

r0
r0v

```
ld r6, r0  //r6<-M[0x21]
inc r0, r0
ld r7, r0  //r7<-M[0x22]
// Now loop forever
ldi r0, 0
// The address of this brz is the one used in validation: Address
    0x19.
brz r0, 0
// You are permitted to add code after this point
// CHANGE HERE and beyond
// Your last instruction should be a jump to location 6
// in order to read the variables into registers r1-r7.
// address 0x1A

// Register Uses
// r0: General Purpose
// r1: Data A (Input)
// r2: Data B (Input)
// r3: Data C (Output)
// r4: General Purpose
// r5: General Purpose
// r6: General Purpose
// r7: Output Address

// Memory Uses
// 0x00: Initial Address A
// 0x01: BCD Pair Count
// 0x02: Address A
// 0x03: Address B
// 0x04: Address C
// 0x05: BCD Pair Count Current

//INITIALISATION  26
ldi  r0, 5          //
shl  r0, r0         //
shl  r0, r0         // 0x14
ldi  r1, 6          //
shl  r1, r1         //
shl  r1, r1         //
inc  r1, r1         //
shl  r1, r1         //
shl  r1, r1         //
shl  r1, r1         //
shl  r1, r1         //
inc  r1, r1         //
```

```
shl  r1, r1          //
shl  r1, r1          //
inc  r1, r1          //
shl  r1, r1          //
shl  r1, r1          //
shl  r1, r1          //
inc  r1, r1          //
shl  r1, r1          //
inc  r1, r1          // 0x6453
st   r0, r1          // ID saved to Memory Address 0x14
ldi  r4, 1           //
ld   r4, r4          // Initialise BCD Pair Counter
ldi  r5, 0           //
ld   r5, r5          // Initialise Address A
ldi  r7, 4           // 04 | Prime Offset
shl  r7, r7          // 08 |
shl  r7, r7          // 10 |
add  r6, r7, r5      // Initialise Address B
add  r7, r7, r6      // Initialise Address C

ldi  r1, 5           // Prime Jump Value: LOAD
shl  r1, r1
shl  r1, r1
shl  r1, r1
shl  r1, r1


//STOREADDR 62
ldi  r0, 2           //
st   r0, r5          // Address A
ldi  r0, 3           //
st   r0, r6          // Address B
ldi  r0, 4           //
st   r0, r7          // Address C
ldi  r0, 5           //
st   r0, r4          // BCD Count
jmp  r1              // Return

//LOADADDR  71
ldi  r0, 2           //
ld   r5, r0          // Address A
ldi  r0, 3           //
ld   r6, r0          // Address B
ldi  r0, 4           //
ld   r7, r0          // Address C
```

```
ldi  r0, 5          //
ld   r4, r0         // BCD Count
jmp  r1             // Return

//LOAD 80
ld   r1, r5         // Load Data A
ld   r2, r6         // Load Data B

//VALIDATION 82
mova r3, r1         // Evaluate A
ldi  r0, 0          // Skip B Evaluation: VAL0
brz  r0, 2          //
//VALB 85
mova r3, r2         //
//VAL0 86
ldi  r5, 4          // 4 |
shl  r5, r5         // 8 |
inc  r5, r5         // 9 |
ldi  r6, 0          //
not  r6, r6         // Mask Established

ldi  r0, 3          // Start Counter
not  r0, r0         //

//VAL1 93
shl  r6, r6         // Shifts Mask 1 Character
shl  r6, r6         //
shl  r6, r6         //
shl  r6, r6         //
not  r6, r6         //
and  r4, r6, r3     // Masks BCD
sub  r4, r5, r4     // Negative if illegal characters are present
not  r6, r6         //
shl  r5, r5         // Shift 1 BCD Character
shl  r5, r5         //
shl  r5, r5         //
shl  r5, r5         //
brn  r4, 13         // WRITEERROR

inc  r0, r0         //
brn  r0, -14        // VAL1

xor  r3, r3, r2
brz  r3  21         // Branch to addition: ADDITION
ldi  r5  5          // Evaluate B: VALB
```

```
174 shl  r5, r5          //
175 shl  r5, r5          //
176 inc  r5, r5          //
177 shl  r5, r5          //
178 shl  r5, r5          //
179 inc  r5, r5          //
180 jmp  r5              //
181
182 //WRITEERROR 118
183 ldi  r1, 0           // Prime 0xFFFF
184 not  r1, r1          //
185 st   r1, r5          // Write 0xFFFF to Memory
186
187 ldi  r2, 3           //
188 shl  r2, r2          //
189 shl  r2, r2          //
190 shl  r2, r2          //
191 inc  r2, r2          //
192 shl  r2, r2          //
193 shl  r2, r2          //
194 shl  r2, r2          //
195 jmp  r2              // Prime Jump: INCINDEX
196
197 //ADDITION 130
198 ldi  r6, 0           //
199 not  r6, r6          // Mask Established
200 shl  r6, r6          // Shifts Mask 1 Character
201 shl  r6, r6          //
202 shl  r6, r6          //
203 shl  r6, r6          //
204
205 ldi  r0, 4           // Character Counter
206 not  r0, r0          //
207 //ADD0 138
208 ldi  r7, 4           // Character Counter
209 not  r7, r7          //
210 sub  r7, r7, r0      // Set count to current position
211 not  r6, r6          //
212 and  r4, r6, r1      // Masks BCD A
213 and  r5, r6, r2      // Masks BCD B
214
215 add  r3, r4, r5      // Adds the masks
216 and  r4, r3, r6      // Masks the Output
217
218 ldi  r5, 4           // 4 |
```

```
shl  r5, r5          // 8 |
inc  r5, r5          // 9 |
//ADDS 149
shl  r5, r5          //
shl  r5, r5          //
shl  r5, r5          //
shl  r5, r5          //
inc  r7, r7          //
brn  r7, -5          // Loops to ADDS

sub r5, r4, r5       // Negative in presence of invalid Values
brn r5, 16           // if negative goto CARRYFIX

//ADDMASK 157
not  r6, r6          //
shl  r6, r6          // Shifts Mask 1 Character
inc  r6, r6          //
shl  r6, r6          //
inc  r6, r6          //
shl  r6, r6          //
inc  r6, r6          //
shl  r6, r6          //
inc  r6, r6          //
not  r6, r6          //
and  r5, r4, r6      // Masks around carry bit
add  r2, r2, r5      // Merge carry bit into addition
not  r6, r6          //

ldi r5, 0            // goto ADD1
brz r5, 14           //
//CARRYFIX 172
ldi  r7, 4           //
not  r7, r7          //
sub  r7, r7, r0      // Reset counter
ldi  r5, 6           //
//CARRYLOOP 176
shl  r5, r5          //
shl  r5, r5          //
shl  r5, r5          //
shl  r5, r5          //
inc  r7, r7          //
brn  r7, -5          // Loops to CARRYLOOP
add  r4, r4, r5      // Add 6 to character
ldi  r7, 0           //
brz  r7, -27         // goto ADDMASK
```

```
264
265 //ADD1 185
266 inc  r0, r0          //
267 brn  r0, 3           //
268 ldi  r4, 0           //
269 brz  r4, 9           //
270 ldi  r4, 7           //
271 shl  r4, r4          //
272 inc  r4, r4          //
273 shl  r4, r4          //
274 shl  r4, r4          //
275 shl  r4, r4          //
276 shl  r4, r4          //
277 jmp  r4             // goto THE_LINKAGE
278
279 ldi  r7, 4           //
280 ld   r7, r7          //
281 st   r7, r3          // Store Output Values
282
283 //INCINDEX 200
284 ldi  r2 4            // Prime Jump LOADADDR
285 shl  r2, r2          //
286 shl  r2, r2          //
287 inc  r2, r2          //
288 shl  r2, r2          //
289 inc  r2, r2          //
290 shl  r2, r2          //
291 inc  r2, r2          //
292
293 ldi  r1 6            // Prime Return INC1
294 shl  r1, r1          //
295 inc  r1, r1          //
296 shl  r1, r1          //
297 shl  r1, r1          //
298 inc  r1, r1          //
299 shl  r1, r1          //
300 inc  r1, r1          //
301 shl  r1, r1          //
302 jmp  r2             //
303
304 //INC1 218
305 inc  r5, r5          // Address A
306 inc  r6, r6          // Address B
307 inc  r7, r7          // Address C
308 dec  r4, r4          // BCD Pair Count
```

```
brz  r4 13          // TERMINATION

ldi  r2, 7          // Prime Jump: STOREADDR
shl  r2, r2         //
inc  r2, r2         //
shl  r2, r2         //
inc  r2, r2         //
shl  r2, r2         //

ldi  r1, 5          // Prime Loop: LOAD
shl  r1, r1         //
shl  r1, r1         //
shl  r1, r1         //
shl  r1, r1         //
jmp  r2             //


//TERMINATION 235
ldi  r0, 6          //
jmp  r0             //

//THE_BUFFER 237
mova r0, r0         // Killing space
mova r0, r0         // to fix a type earlier in the design.
mova r0, r0         //

//THE_LINKAGE 240
ldi  r4, 4          //
shl  r4, r4         //
shl  r4, r4         //
inc  r4, r4         //
shl  r4, r4         //
shl  r4, r4         //
inc  r4, r4         //
shl  r4, r4         //
jmp  r4             // goto ADD0
```

Appendix A: Simulation Waveforms