ShmooCon 8 Badge Contest
Badge Data / Disassembly
Darth Null, Feburary 2012

Instructions partially disassembled by my script, partially corrected and tweaked after review of original source code (graciously provided by Adam). Certainly not intended to be a readable / reuseable source file, but instead to show what each and every row of punches on the badges meant.

Some notes:

(xx) indicates an indirect memory access.
That is, don't load what's in xx, but load what's in the memory address that's stored in xx.

Memory locations 10-17 are auto-increment "registers" - as far as I can tell, every time you read from them the computer increments the value that's stored there.

When calling a function, the PC for the next instruction (after the call) is stored at the address of the function. So to return, the function does an indirect jump to the address stored at the top of the fn.

```
 +-- Memory Location (octal)
 |       +-- punches / dots
 |       |         +-- opcode (octal)  +-- notes, etc.
 |       |         |    + ASCII        |
 v  vvvvvvvvvvvv  vvvv  v   vvvvvvvvvvvvvvvvv....

Badge 1     ||||||          [badge number and I Ching label]

022 .....*..**..  0114  L   # "LD 22 GO 75           "
023 .....*...*..  0104  D   # instructions to operator
024 ......*.....  0040      # load program into memory starting at
025 ......**..*.  0062  2   # location 22 (octal), then begin execution
026 ......**..*.  0062  2   # at location 75
027 ......*.....  0040
030 .....*...***  0107  G
031 .....*..****  0117  O
032 ......*.....  0040
033 ......**.***  0067  7
034 ......**.*.*  0065  5
035 ......*.....  0040
036 ......*.....  0040
037 ......*.....  0040
040 ......*.....  0040
041 ......*.....  0040
042 ......*.....  0040
043 ......*.....  0040
044 ......*.....  0040
045 ......*.....  0040
```

```
046 ......*.....  0040
047 ......*.....  0040   ##### Lookup table for strings, functions follows ##
050 ....*.*..*..  0244 [orig Lucifer decrypt]
051 ....*...***.  0216 [Decrypt function]
052 .....*.***..  0134 [Get Char function]
053 .....**...**  0143 [Get Password function]
054 .....***.**.  0166 [Print String function]
055 .....*****.*  0175 [Print Char function]
056 ...*......*.  0402 [Enigma settings string]
057 ...*..*..***  0447 [Enigma 1st ciphertext line string]


Badge 2     ::::::

060 ...*.*.*..**  0523 [Enigma 2nd ciphertext line string]
061 ....**..*.*.  0312 [SBoxes string]
062 ....*....*.*  0205 [Xor function]
063 ....*.....**  0203 [Xor parameter A]
064 ....*....*..  0204 [Xor parameter B]
065 ....**.*..*.  0322 [Passwort string]
066 ....***..**.  0346 [Password buffer]
067 ...**....*.*  0605 [Final CT string]
070 ...**.**....  0660 [Final decrypt buffer]
071 ....**......  0300 [unused char buffer from abandoned Lucifer impl.]
072 ....*.*****.  0276 [Current CT char buffer]
073 ....*.******  0277 [Current pass char buffer]
074 ...**.......  0600 [CR / LF string]
075 ***.**......  7300     cla cll      # **** BEGIN PROGRAM HERE **** (clr AC)
076 **.....**.*.  6032     IOT  32      # enable / clear keyboard
077 **....*..**.  6046     IOT  46      # enable / clear output
100 ***.**......  7300     cla cll
101 ..**..*.***.  1456     tad  (56)    # load AC with addr of enigma settings
102 *..*..*.**..  4454     jms  (54)    # call print string
103 ***.**......  7300     cla cll
104 ..**..*.****  1457     tad  (57)    # enigma 1st line
105 *..*..*.**..  4454     jms  (54)    # call print string
106 ***.**......  7300     cla cll
107 ..**..**....  1460     tad  (60)    # enigma 2nd line
110 *..*..*.**..  4454     jms  (54)    # call print string
111 ***.**......  7300     cla cll
112 ..**..**.*.*  1465     tad  (65)    # "Passwort"
113 *..*..*.**..  4454     jms  (54)    # call print string
114 ***.**......  7300     cla cll
115 *..*..*.*.**  4453     jms  (53)    # call get password

Badge 3     |:::|:

116 ***.**......  7300     cla cll      # clear accumulator, L
117 ..**..***...  1470     tad  (70)    # first char of final msg decrypt buf
120 .**.....*..*  3011     dca  11      # store into auto-inc reg 11
121 ..**..**.***  1467     tad  (67)    # first char of final ciphertext
122 .**.....**..  3014     dca  14      # store into auto-inc reg 14
123 ..**..**.**.  1466     tad  (66)    # first char of temp passwd storage
124 *..*..*.*..*  4451     jms  (51)    # call decrypt
```

```
125 ***.**......  7300     cla cll
126 ..**..****..  1474     tad  (74)    # cr/lf
127 *..*..*.**..  4454     jms  (54)    # call print string
130 ***.**......  7300     cla cll
131 ..**..***...  1470     tad  (70)    # final msg (now decrypted)
132 *..*..*.**..  4454     jms  (54)    # print it
133 ****......*.  7402     hlt          # halt
134 ............  0000  #####   GetChar        #####
135 **.....**..*  6031     IOT  31      # skip next if there's a key ready
136 *.*..*.***.*  5135     jmp  135     # no, loop until...
137 **.....****.  6036     IOT  36      # read char from keyboard into AC
140 *.**.*.***..  5534     JMP  (134)   # return to caller
141 ........**.*  0015  \r              # handy carriage return
142 ............  0000                  # single passwd char buffer
143 ............  0000  #####   Get Password   #####
144 ***.**......  7300     cla cll
145 ..**..**.**.  1466     tad  (66)    # store address of password buffer
146 .**.....****  3017     dca  17      #   in auto-inc reg 17
147 *..*..*.*.*.  4452     jms  (52)    # get a char
150 .**..**...*.  3142     dca  142     # move it to 0x142
151 ..*..**...*.  1142     tad  142     # and grab back a copy
152 ***...*....*  7041     cma  iac     # negate / increment AC  (AC *= -1)
153 ..*..**....*  1141     tad  141     # add the handy CR
```

Badge 4 :|:::|

```
154 ****..*.*...  7450     sna          # skip if AC != 0 (char != CR)
155 *.*..***.*.*  5165     jmp  165     # else skip to fn return (they hit cr)
156 ***.**......  7300     cla cll      # clear AC
157 ..*..**...*.  1142     tad  142     # grab the key again
160 *..*..*.**.*  4455     jms  (55)    # echo it to the user
161 ***.**......  7300     cla cll      # clear
162 ..*..**...*.  1142     tad  142     # load it again (AC mod'd by print char)
163 .***....****  3417     dca  (17)    # and store it in next loc in buffer
164 *.*..**..***  5147     jmp  147     # loop back to get the next char
165 *.**.**...**  5543     jmp  (143)   # return to caller
166 ............  0000  #####   Print String   #####
167 .**.....*...  3010     dca  10      # string loc -> auto-inc reg 10
170 ..**....*...  1410     tad  (10)    # first char from string -> AC
171 ****..*.*...  7450     sna          # skip next if AC != 0
172 *.**.***.**.  5566     jmp  (166)   # return to caller
173 *..*..*.**.*  4455     jms  (55)    # call print char
174 *.*..****...  5170     jmp  170     # loop back and do it again
175 ............  0000  #####   Print Char     #####
176 **....*....*  6041     IOT  41      # skip next if output device ready
177 *.*..******.  5176     jmp  176     # otherwise loop until ready
200 **....*..**.  6046     IOT  46      # print AC to output
201 ***.**......  7300     cla cll      # clear things
202 *.**.*****.*  5575     jmp  (175)   # return to caller
203 ............  0000                  # XOR parm A
204 ............  0000                  # XOR parm B
205 ............  0000  #####   XOR           ##### [ (A+B) - 2*(A&B) ]
206 ***.**......  7300     cla cll
207 ..*.*.....**  1203     tad  203     # load A in AC
```

```
210 ....*....*..  0204     and  204     # bitwise and with B (AC = A&B)
211 ***...*....*  7041     cma iac      # negate AC
```

Badge 5 |||:|:

```
212 ***..*...*..  7104     cll ral      # clear L, rotate L,AC 1 (mult by 2)
213 ..*.*.....**  1203     tad  203     # add A
214 ..*.*....*..  1204     tad  204     # add B
215 *.***....*.*  5605     jmp  (205)   # return to caller, result in AC
216 ............  0000  #####   Decrypt      #####
217 .**.....*...  3010     dca  10      # loc of passwd buf into reg 10
220 ..**....**..  1414     tad  (14)    # next char of ciphertext -> AC
221 ****..*.*...  7450     sna          # skip next if AC != 0
222 *.***...***.  5616     jmp  (216)   # return if AC == 0 (end of string)
223 .**...***.*.  3072     dca  72      # hold current CT char in buffer
224 ..**....*...  1410     tad  (10)    # load current key char -> AC
225 ****..*.....  7440     sza          # skip next if AC == 0
226 *.*.*..**.**  5233     jmp  233     # jump ahead, not at end of passwd
227 ***.**......  7300     cla cll
230 ..**..**.**.  1466     tad  (66)    # back to beginning of password
231 .**.....*...  3010     dca  10      # re-store in auto-inc reg 10
232 ..**....*...  1410     tad  (10)    # and pull pw char back into AC again
233 .***..**..**  3463     dca  (63)    # pwd char -> XOR parm A
234 ..*...***.*.  1072     tad  72      # grab current CT char
235 .***..**.*..  3464     dca  (64)    # CT -> XOR parm B
236 *..*..**..*.  4462     jms  (62)    # call XOR function
237 .**...***.*.  3072     dca  72      # store result (PW^CT) into buffer
240 ***.**......  7300     cla cll
241 ..*...***.*.  1072     tad  72      # and pull it back into AC
242 .***....*..*  3411     dca  (11)    # ..add to output buf (see instr 120)
243 *.*.*..*....  5220     jmp  220     # loop back and do it again
244 ............  0000  ##### original decrypt func (unused) #####
245 .**.....*...  3010     dca  10      # loc of passwd buf into reg 10
246 ..**....*...  1410     tad  (10)    # read first char of pwd into AC
247 .**...***.**  3073     dca  73      # and store it in 73
```

Badge 6 :|:|||

```
250 ..**..***...  1470     tad  (70)    # loc of decrypt buffer
251 .**.....*..*  3011     dca  11      # store in reg 11
252 ..**....**..  1414     tad  (14)    # next char of ciphertext -> AC
253 ****..*.*...  7450     sna          # skip next if AC != 0
254 *.***.*..*..  5644     jmp  (244)   # else return to caller (end of str)
255 .**...***.*.  3072     dca  72      # store CT char in buffer
256 ..*...***.**  1073     tad  73      # retrieve the passwd char
257 ****..*.....  7440     sza          # skip next if AC == 0
260 *.*.*.**.*.*  5265     jmp  265     # jump ahead, not at end of passwd
261 ***.**......  7300     cla cll
262 ..**..**.**.  1466     tad  (66)    # return to beginning of password buf
263 .**.....*...  3010     dca  10      # and store pointer in reg 10
264 ..**....*...  1410     tad  (10)    # and pull current pw char back
265 .**...***.**  3073     dca  73      # shove it into storage
266 ..*...***.*.  1072     tad  72      # get CT char from storage
267 ..*...***.*.  1072     tad  72      # and again [Sblock call del'd frm src]
```

```
270 .***..**..**  3463     dca  (63)    # CT char -> XOR parm A
271 ..*...***.**  1073     tad  73      # get PW char
272 .***..**.*..  3464     dca  (64)    # PW char -> XOR parm B
273 *..*..**..*.  4462     jms  (62)    # call XOR function
274 .***....*..*  3411     dca  (11)    # and store result in output buf
275 *.*.*.*.*.*.  5252     jmp  252     # loop back and do it again
276 ............  0000  ##### originally, 2 words of buffer, and
277 ............  0000  ##### a function to implement s-boxes from Lucifer
300 ............  0000  ##### now, replaced with an inside joke
301 .....*.*..*.  0122  R     # R1JND43L?
302 ......**...*  0061  1
303 .....*..*.*.  0112  J
304 .....*..***.  0116  N
305 .....*...*..  0104  D
```

Badge 7 :|::::

```
306 ......**.*..  0064  4
307 ......**..**  0063  3
310 .....*..**..  0114  L
311 ......******  0077  ?
312 ...*.***...*  0561  ##### next 8 words are merged s-box data #####
313 .*..**...***  2307
314 *.*.*...**..  5214
315 .******.*...  3750
316 ..***..**.*.  1632
317 ****..*...**  7443
320 *......*.**.  4026
321 **...*****.*  6175  ##### end s-boxes #####
322 ....**.*..*.  0322      # [password prompt]
323 .....*.*....  0120  P
324 .....**....*  0141  a
325 .....***..**  0163  s
326 .....***..**  0163  s
327 .....***.***  0167  w
330 .....**.****  0157  o
331 .....***..*.  0162  r
332 .....***.*..  0164  t
333 ......*.....  0040
334 .....**..*.*  0145  e
335 .....**.*..*  0151  i
336 .....**.***.  0156  n
337 .....**..***  0147  g
340 .....**..*.*  0145  e
341 .....**...*.  0142  b
342 .....**..*.*  0145  e
343 .....**.***.  0156  n
```

Badge 8 ::::|:

```
344 ......***.*.  0072  :
345 ............  0000
346 ....***..**.  0346      # temp storage for password from user
347 ............  0000  .
```

```
350 ............  0000  .
351 ............  0000  .
352 ............  0000  .
353 ............  0000  .
354 ............  0000  .
355 ............  0000  .
356 ............  0000  .
357 ............  0000  .
360 ............  0000  .
361 ............  0000  .
362 ............  0000  .
363 ............  0000  .
364 ............  0000  .
365 ............  0000  .
366 ............  0000  .
367 ............  0000  .
370 ............  0000  .
371 ............  0000  .
372 ............  0000  .
373 ............  0000  .
374 ............  0000  .
375 ............  0000  .
376 ............  0000  .
377 ............  0000  .
400 ............  0000  .
401 ............  0000  .
```

Badge 9 |||:||

```
402 ...*......*.  0402     # Enigma Settings
403 .....*.*..*.  0122  R  # R:TMW B3M W:412 M:HAK P:S->H M->O
404 ......***.*.  0072  :
405 .....*.*.*..  0124  T
406 .....*..**.*  0115  M
407 .....*.*.***  0127  W
410 ......*.....  0040
411 .....*....*.  0102  B
412 ......**..**  0063  3
413 .....*..**.*  0115  M
414 ......*.....  0040
415 .....*.*.***  0127  W
416 ......***.*.  0072  :
417 ......**.*..  0064  4
420 ......**...*  0061  1
421 ......**..*.  0062  2
422 ......*.....  0040
423 .....*..**.*  0115  M
424 ......***.*.  0072  :
425 .....*..*...  0110  H
426 .....*.....*  0101  A
427 .....*..*.**  0113  K
430 ......*.....  0040
431 .....*.*....  0120  P
432 ......***.*.  0072  :
```

```
433 .....*.*..**  0123  S
434 ......*.**.*  0055  -
435 ......*****.  0076  >
436 .....*..*...  0110  H
437 ......*.....  0040
```

Badge 10 ||:|||

```
440 .....*..**.*  0115  M
441 ......*.**.*  0055  -
442 ......*****.  0076  >
443 .....*..****  0117  O
444 ........**.*  0015  \r
445 ........*.*.  0012  \n
446 ............  0000
447 ...*..*..***  0447     # Enigma ciphertext line 1
450 .....*..**.*  0115  M  # MICCK FJBZG DNQIH HYDNB CTKTN KLLTR FCJEJ HBZUL
451 .....*..*..*  0111  I
452 .....*....**  0103  C
453 .....*....**  0103  C
454 .....*..*.**  0113  K
455 .....*...**.  0106  F
456 .....*..*.*.  0112  J
457 .....*....*.  0102  B
460 .....*.**.*.  0132  Z
461 .....*...***  0107  G
462 .....*...*..  0104  D
463 .....*..***.  0116  N
464 .....*.*...*  0121  Q
465 .....*..*..*  0111  I
466 .....*..*...  0110  H
467 .....*..*...  0110  H
470 .....*.**..*  0131  Y
471 .....*...*..  0104  D
472 .....*..***.  0116  N
473 .....*....*.  0102  B
474 .....*....**  0103  C
475 .....*.*.*..  0124  T
```

Badge 11 |||:::

```
476 .....*..*.**  0113  K
477 .....*.*.*..  0124  T
500 .....*..***.  0116  N
501 .....*..*.**  0113  K
502 .....*..**..  0114  L
503 .....*..**..  0114  L
504 .....*.*.*..  0124  T
505 .....*.*..*.  0122  R
506 .....*...**.  0106  F
507 .....*....**  0103  C
510 .....*..*.*.  0112  J
511 .....*...*.*  0105  E
512 .....*..*.*.  0112  J
```

```
513 .....*..*...  0110  H
514 .....*....*.  0102  B
515 .....*.**.*.  0132  Z
516 .....*.*.*.*  0125  U
517 .....*..**..  0114  L
520 ........**.*  0015  \r
521 ........*.*.  0012  \n
522 ...........   0000
523 ...*.*.*..**  0523     # Enigma ciphertext line 2
524 .....*.*....  0120  P  # PUFII DLCPJ XXSRK HNLPP YJCUY XGTGD FUWDY JEKZD Y
525 .....*.*.*.*  0125  U
526 .....*...**.  0106  F
527 .....*..*..*  0111  I
530 .....*..*..*  0111  I
531 .....*...*..  0104  D
532 .....*..**..  0114  L
533 .....*....**  0103  C
```

Badge 12 :::|||

```
534 .....*.*....  0120  P
535 .....*..*.*.  0112  J
536 .....*.**...  0130  X
537 .....*.**...  0130  X
540 .....*.*..**  0123  S
541 .....*.*..*.  0122  R
542 .....*..*.**  0113  K
543 .....*..*...  0110  H
544 .....*..***.  0116  N
545 .....*..**..  0114  L
546 .....*.*....  0120  P
547 .....*.*....  0120  P
550 .....*.**..*  0131  Y
551 .....*..*.*.  0112  J
552 .....*....**  0103  C
553 .....*.*.*.*  0125  U
554 .....*.**..*  0131  Y
555 .....*.**...  0130  X
556 .....*...***  0107  G
557 .....*.*.*..  0124  T
560 .....*...***  0107  G
561 .....*...*..  0104  D
562 .....*...**.  0106  F
563 .....*.*.*.*  0125  U
564 .....*.*.***  0127  W
565 .....*...*..  0104  D
566 .....*.**..*  0131  Y
567 .....*..*.*.  0112  J
570 .....*...*.*  0105  E
571 .....*..*.**  0113  K
```

Badge 13 |:||||

```
572 .....*.**.*.  0132  Z
```

```
573 .....*...*..  0104  D
574 .....*.**..*  0131  Y
575 ........**.*  0015  \r
576 ........*.*.  0012  \n
577 ............  0000
600 ...**.......  0600  .  # CRLF
601 ........**.*  0015  \r
602 ........*.*.  0012  \n
603 ............  0000
604 ............  0000
605 ...**....*.*  0605     # encrypted final clue
606 ...........*  0001  .  # ^A,!6G1Mg1*>+I)^N(UR4&7^P ;R5,%g5& &=$g4:7"'
607 ......*.**..  0054  ,
610 ......*....*  0041  !
611 ......**.**.  0066  6
612 .....*...***  0107  G
613 ......**...*  0061  1
614 .....*..**.*  0115  M
615 .....**..***  0147  g
616 ......**...*  0061  1
617 ......*.*.*.  0052  *
620 ......*****.  0076  >
621 ......*.*.**  0053  +
622 .....*..*..*  0111  I
623 ......*.*..*  0051  )
624 ........***.  0016  .
625 ......*.*...  0050  (
626 .....*.*.*.*  0125  U
627 .....*.*..*.  0122  R
```

Badge 14 |||:|

```
630 ......**.*..  0064  4
631 ......*..**.  0046  &
632 ......**.***  0067  7
633 ........***.  0016  .
634 ......*.....  0040
635 ......***.**  0073  ;
636 .....*.*..*.  0122  R
637 ......**.*.*  0065  5
640 ......*.**..  0054  ,
641 ......*..*.*  0045  %
642 .....**..***  0147  g
643 ......**.*.*  0065  5
644 ......*..**.  0046  &
645 ......*.....  0040
646 ......*..**.  0046  &
647 ......****.*  0075  =
650 ......*..*..  0044  $
651 .....**..***  0147  g
652 ......**.*..  0064  4
653 ......***.*.  0072  :
654 ......**.***  0067  7
655 ......*...*.  0042  "
```

```
656 ......*..***  0047  '
657 ............  0000
660 ...**.**....  0660     # buffer space to decrypt final clue
661 ......*.....  0040
662 ......*.....  0040
663 ......*.....  0040
664 ......*.....  0040
665 ......*.....  0040
```

Badge 15 ::|:::

```
666 ......*.....  0040
667 ......*.....  0040
670 ......*.....  0040
671 ......*.....  0040
672 ......*.....  0040
673 ......*.....  0040
674 ......*.....  0040
675 ......*.....  0040
676 ......*.....  0040
677 ......*.....  0040
700 ......*.....  0040
701 ......*.....  0040
702 ......*.....  0040
703 ......*.....  0040
704 ......*.....  0040
705 ......*.....  0040
706 ......*.....  0040
707 ......*.....  0040
710 ......*.....  0040
711 ......*.....  0040
712 ......*.....  0040
713 ......*.....  0040
714 ......*.....  0040
715 ......*.....  0040
716 ......*.....  0040
717 ......*.....  0040
720 ......*.....  0040
721 ......*.....  0040
722 ......*.....  0040
723 ......*.....  0040
```

Badge 16 :::|::

```
724 ......*.....  0040
725 ......*.....  0040
726 ......*.....  0040
727 ......*.....  0040
730 ......*.....  0040
731 ......*.....  0040
000 ............  0000  ##### Easter Egg. Load beginning at 75, start there ###
075 ..*..*..*...  1110     tad  110         # beginning of string
076 .**.....*...  3010     dca  10          # store in auto-inc reg 10
077 ..**....*...  1410     tad  (10)        # read first char into AC
```

```
100 ****..*.*...  7450     sna              # skip next if AC != 0
101 *.*..*...***  5107     jmp  107         # else end of string, jump to end
102 **....*....*  6041     IOT  41          # skip next if output not ready
103 *.*..*....*.  5102     jmp  102         # loop until output device ready
104 **....*..**.  6046     IOT  46          # print AC on output
105 ***.**......  7300     cla cll
106 *.*...******  5077     jmp  77          # and loop back to do next char
107 ****......*.  7402     hlt              # done. halt computer.
110 .....*..*...  0110     # Hello World
111 .....*..*...  0110  H
112 .....**..*.*  0145  e
113 .....**.**..  0154  l
114 .....**.**..  0154  l
115 .....**.****  0157  o
116 ......*.....  0040
117 .....*.*.***  0127  W
120 .....**.****  0157  o
121 .....***..*.  0162  r
122 .....**.**..  0154  l
123 .....**..*..  0144  d
```

###