EEL3701 – Dr. Gugel
Spring 2015
Final Quiz

Last Name Key First Name _____

UF ID# _____

Open book/open notes, **90-minute** exam to be done in **non-red** pen. ***No electronic devices permitted.***

Point System: Page 1 => 8 points Veronica

Page 2 => 12 points Madison

Page 3 => 15 points Mitch #5-9 / Alan #10

Page 4 => 15 points Mason #12 & #14 / Jonathan #11

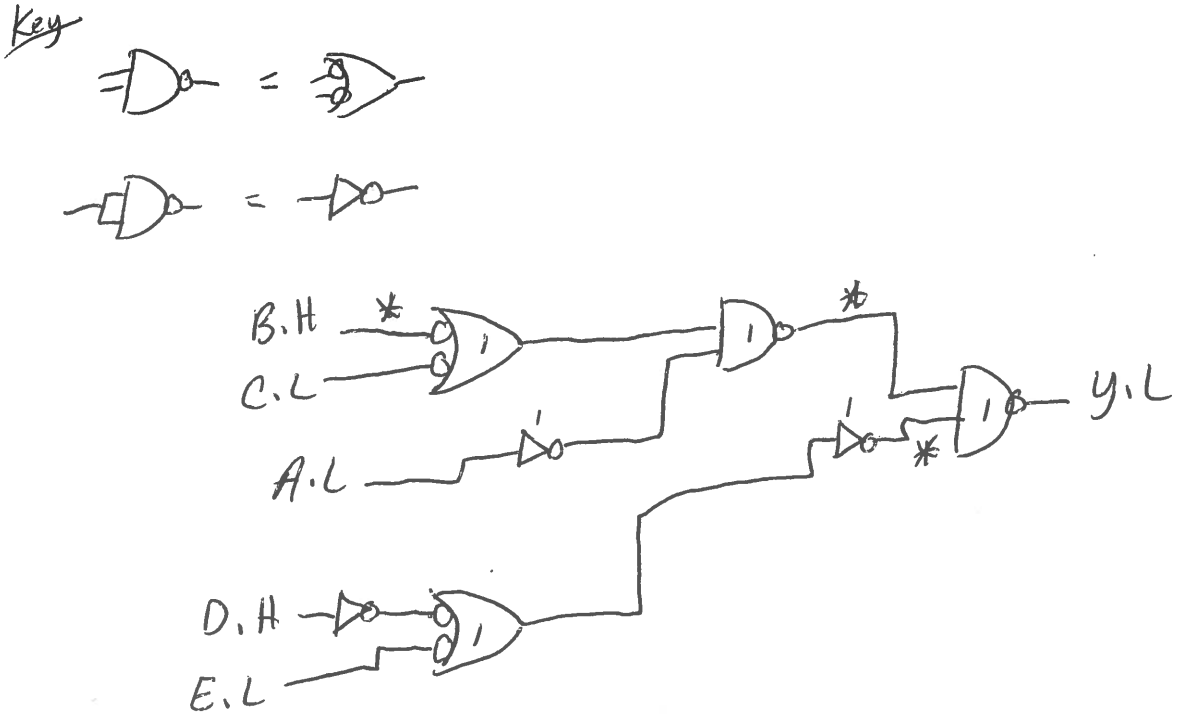
TOTAL => _____ (50 pt.)

Grade Review Information: (*NOTE: deadline of request for grade review is the day the exam is returned.*)

1. Implement the logic equation below using 2 Input NAND Gates only. (6 pt.)

$$Y = \overline{(A*(\overline{B}+C))}\ \overline{(D+E)} \quad ; A.L, B.H, C.L, D.H, E.L, Y.L$$

Key



2. The G GPU has been redesigned to initially boot from **8000 Hex**. i.e. Fetch it's first instruction from **8000H** in memory. Assuming that there is a **low true Reset** signal, explain what needs to be modified in the current design to facilitate booting from **8000 Hex**. (2 pt.) Best Answer = Most Points!

Use Asynchronous set & CLR on FFs that make-up the PC

Specifically, CLR PC14:0 upon reset and set PC15 upon reset.

PC +1 set/CLR +1

Pg. Score =

Lazy Key

3. A student has interfaced a **2K x 8 ROM** with the G-CPU ***starting at address 8000 Hex***. The student also has interfaced and placed ***4K x 8 RAM in the highest 4K of the G-CPU memory map***. What is the memory ranges for the ROM and RAM devices? (2 pt.)

5K 1K = 2^{10} A9:0 1000|00xx|xxxx|xxxx

1000|8xxx|xxxx|xxxx

$2K \rightarrow 2^1 \times 2^{10} = 2^{11}$

ROM Memory Range 8000 Hex to 87FF (93FF) (Hex) correct for 5K

2K 1111|xxxx|xxxx|xxxx

$4K \rightarrow 2^2 \cdot 2^{10} = 2^{12}$

RAM Memory Range F000 (Hex) to FFFF Hex

---

4. A vector stored in RAM consists of **signed numbers that are one byte in length**. Write the code to convert all the negative numbers to positive numbers. Assume **B contains the length** of the vector and **Y contains the starting address** of the vector. Also, if you need a counter in memory, use address **FFFF Hex**. (10 pt.)

```
STAB $FFFF ; save counter in memory
TOP: Ldaa 0,y ; get value
(1)  BP   SKIP
(1)  Ldab #$FF , B=-1
     sum_BA ; Neg. no -1
(1)  ComA ; complement A
(1)  staa 0,y ; store pos. no.
(1)  INY ; INC PTR
(1)  SKIP: Ldaa $FFFF ; get count
(1)       sum_BA ; count = count -1
(1)       BEQ  DONE ; check if zero
(1)       STAA $FFFF ; save counter
          BNE  TOP
DONE: Ldaa #0 ; infinite loop
      BEQ  DONE
```

1 pt for both

Madison

if you have a better scheme use it and document for later

Page Score =

Key

For the next set of questions, consult the G-CPU code in **Appendix A** and assume the following initial conditions: **A = 5, X = $FF00**, Memory location ***$FF00 = $00***, **$FF01 = $11, $FF02 = $22 .... $FF08 = $88.**

5. What is the effective address for the STAB instruction at $8006? FFFE Hex (2 pt.)

6. What is the effective address for the LDAB instruction at $8009? 8000A Hex (2 pt.)

7. How many times is the LDAA 0,X at address $801C executed at run time? 2 Hex (2 pt.)

8. What values in SRAM memory below when the PC is fetching $8028? (3 pt.)

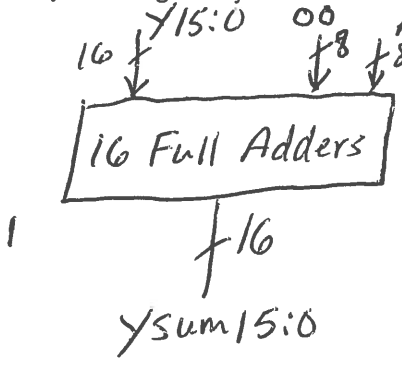
0.5 $FF00 = 44 Hex 0.5 $FF01 = 33 Hex 0.5 $FF02 = 22 Hex

0.5 $FF03 = 11 Hex 0.5 $FF04 = 00 Hex 0.5 $FF05 = 55 Hex

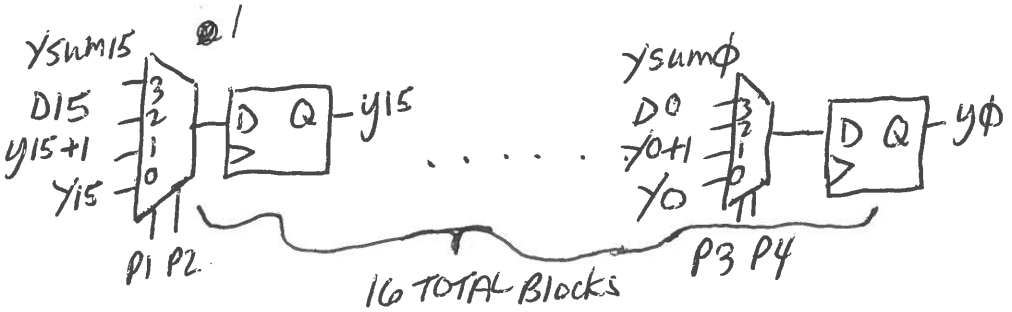
9. What is the function of this short program? Reverse the elements in the vector.
i.e. 1st ↔ last, 2nd ↔ 2nd from last, etc. (2 pt.)

---

10. Using only 2:1 decoders, flip-flops, 2:1 muxes, single bit full adders and any other elementary digital components, design the hardware required to implement the SUM_AY instruction. Label all buses and explain any new signals you'll need in the ALU, Controller, IR and/or Address Mux. Best design = Most points. (4 pt.)



Mux "3" = sumAY, "2" = load data, "1" = INC y, "0" = hold



YLDL	YINC	YAY	P1 P2	
0	0	0	0 0	hold
0	0	1	1 1	sum AY
0	1	0	0 1	INC y
1	0	0	1 0	load data

2

$P1 = \overline{yINC}(YLDL \oplus yAY)$

$P2 = \overline{YLDL}(YINC \oplus yAY)$

↑ needs Y_LDL

$P3 = \overline{yINC}(yLDL \oplus yAY)$

$P4 = \overline{YLDL}(yINC \oplus yAY)$

↑ needs Y_LDL

Page Score =

11. For the G CPU code shown in Appendix A, fill in the cycle table below for the ***first pass*** for the lines of code shown below. ***Use the initial conditions shown on the previous page and show all answers in Hex.*** (9 pt.)

```
LDAB    $FFFE    ; first pass of this instruction
DECA             ; assume this takes 2 cycles to execute
```

IR = 6 bits Max IR = 3F

Cycle	Controller State (1)	Addr Bus (2)	Data Bus (1.5)	IR (1)	R/-W (0.5)	B Reg (1)	Dev driving Addr Bus (1)	Dev Driving Data Bus (1)
1	0	8012	05	04	1	01	PC	ROM
2	1	8012	05	05	1	01	PC	ROM
3	7	8013	FE	05	1	01	PC	ROM
4	8	8014	FF	05	1	01	PC	ROM
5	9	FFFE	FD	05	1	01	MAR	RAM
6	0	8015	New Opcode	05	1	FD	PC	ROM
7	1	8015	New Opcode	New opcode	1	FD	PC	ROM

New Opcode = 24 to 2F or 32 to 3F

16K

12. The G CPU Next State Table and Controller can be found in **Appendix B**. The Controller was created via an ~~8K~~ **x 32 ROM**. For the very ***first*** and ***last*** cycles of the **LDAB $FFFE** instruction above, show the addresses and corresponding data that **need to be programmed in the ROM**. (4 pt.)

First Cycle Executed

ROM Addresses = 0 - 00FF Hex Data = 058103 Hex

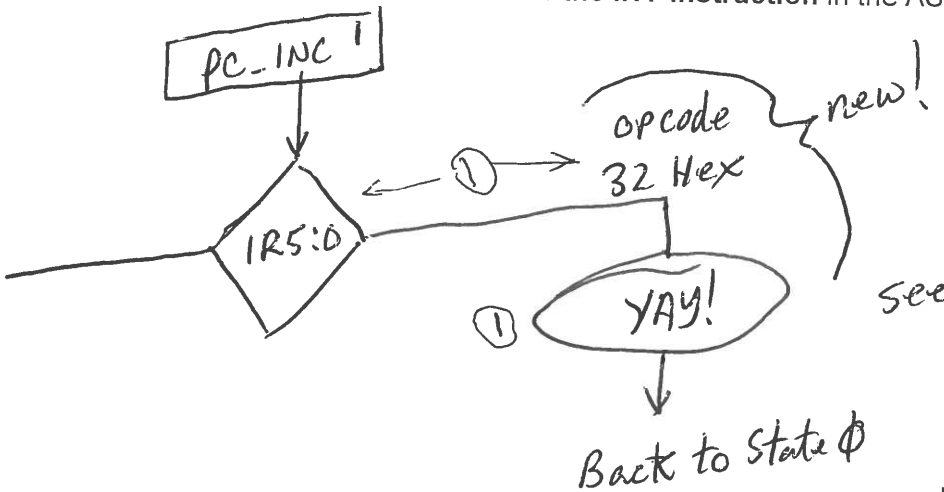
Upper (3) Bytes Only! → Logic 058300

Last Cycle Executed

ROM Addresses = 0900 - 09FF Hex Data = 0307FC Hex

Lower (3) Bytes Only! → Logic 010400

14. For the new **SUM_AY** instruction in **Appendix A**, show below what needs to be **added** to the Controller **ASM Flow Chart** assuming that this instruction will **follow the INY instruction** in the ASM Chart. (2 pt.)

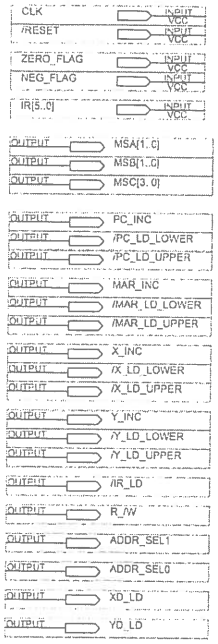


see previous page for signal YAY

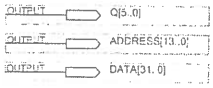
Page Score =

Appendix B. G-CPU Controller & Next State Table:

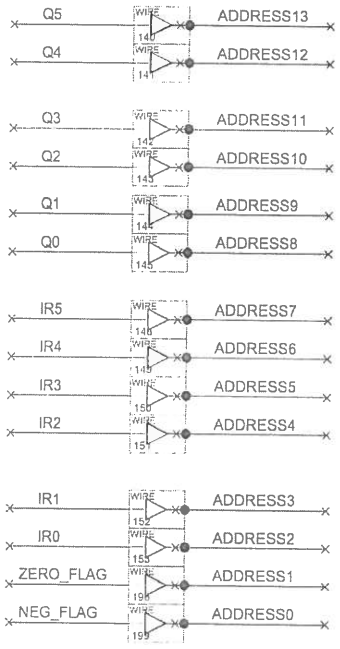
Controller Input & Output



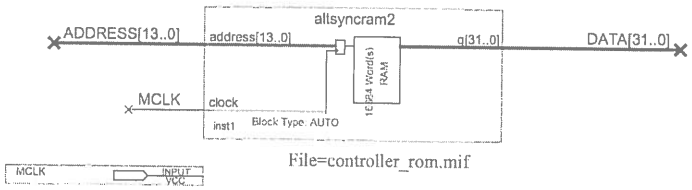
Debug Purposes



Controller Inputs (State, Flags, Instruction)

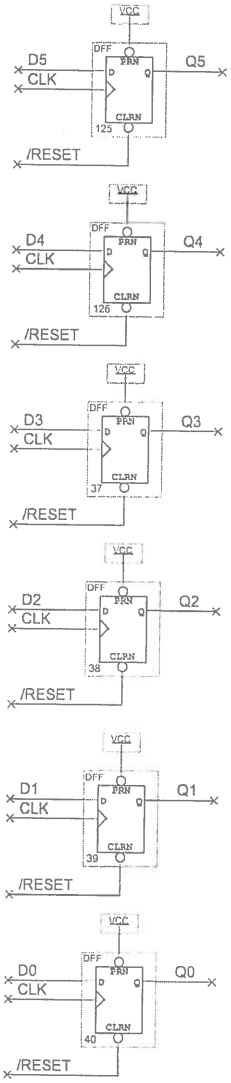


Controller Logic

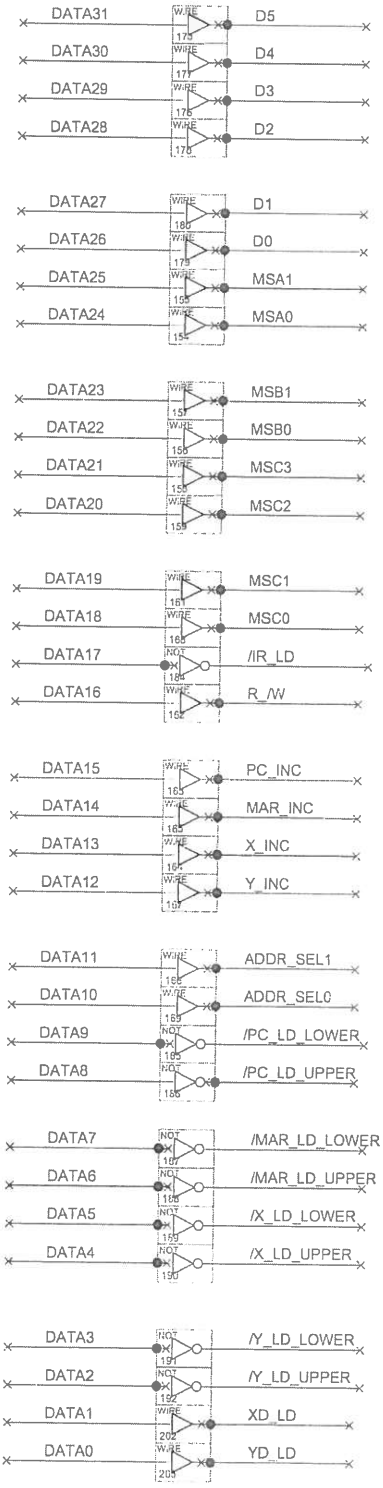


File=controller_rom.mif

ASM State Generation



Controller Outputs



Page Score =

Appendix A. G-CPU Code (address of instruction in memory shown on left):

Register ***A = Vector Length***, Register ***X = Starting Address of Vector***

X = $FF00 A = 5

```
                        ORG     $8000
8000 ✓  Compute:        TXY     y = FF00    ;new instruction transfer X => Y
8001 ✓                  SUM_AY  y = FF05    ;new instruction A + Y => Y
8002 ✓                  DEY     y = FF04    ;new instruction Decrement Y
8003 ✓                  SHFA_R  A = 2
8004 ✓                  TAB     B = 2
8005 ✓                  COMB    B = FD
8006,7,8 ✓              STAB    $FFFE       FFFE[FD]
8009,A ✓                LDAB    #1  B = 1
800B ✓                  SUM_BA  A = 3
800C,D,E ✓              STAA    $FFFF       FFFF[3]     2nd pass

800F,10,11 ✓ Loop:      LDAA    $FFFF  A = 3  ) A = 2
8012,13,14 ✓            LDAB    $FFFE  B = FD
8015 ✓                  DECA    A = 2 , A = 1   ;new instruction Decrement A
8016 ✓                  TAB     B = 2           2nd Pass
8017,18 ✓               BEQ     Done
8019 ✓ 1A,1B ✓          STAA    $FFFF       FFFF[2]
801C,1D ✓               LDAA    0,X         A = 00
801E,1F ✓               LDAB    0,Y         B = 44
8020,21 ✓               STAA    0,Y         FF04[00]
8022,23 ✓               STAB    0,X         FF00[44]
8024 ✓                  INX     X = FF01
8025 ✓                  DEY     y = FF03    ;new instruction, Decrement Y
8026,27 ✓               BRA     Loop        ;new instruction, unconditional branch
8028 ✓  Done:           BRA     Done        ;new instruction, unconditional branch
```

1101

Page Score =

0 1 1 0 1 4 1 0 1 ?

0000 / 0001 / 0000 / 0100 / 0000 / 0000

G-CPU Controller Next State Table

Pres State	Opcode	Flags	Next State	Mux Select			Control		REG INC	ADDR SEL	PC	MAR	X,Y Loading		Disp Regs	
Q[5..0]	IR[5..0]	Z N	D[5..0]	MSA [1..0]	MSB [1..0]	MSC [3..0]	IR LD	R /W	PC MAR X Y	ADDR SEL [1..0]	PC LD L/U	MAR LD L/U	X LD L/U	Y LD L/U	XD_LD YD_LD	Present State Function
000000	XXXXXX	XX	000001	01	10	0000	1	1	0000	00	00	00	00	00	00	generic instruction fetch
000001	000000	XX	000000	01	01	0000	0	1	1000	00	00	00	00	00	00	Transfer A to B (TAB)
000001	000001	XX	000000	10	10	0000	0	1	1000	00	00	00	00	00	00	Transfer B to A (TBA)
000001	000010	XX	000010	01	10	0000	0	1	1000	00	00	00	00	00	00	LDAA #data, state 1
000010	XXXXXX	XX	000000	00	10	0000	0	1	1000	00	00	00	00	00	00	LDAA #data, state 2
000001	000011	XX	000011	01	10	0000	0	1	1000	00	00	00	00	00	00	LDAB #data, state 1
000011	XXXXXX	XX	000000	01	00	0001	0	1	1000	00	00	00	00	00	00	LDAB #data, state 3
000001	000100	XX	000100	01	10	0000	0	1	1000	00	00	00	00	00	00	LDAA addr, state 1
000100	XXXXXX	XX	000101	01	10	0000	0	1	1000	00	00	10	00	00	00	LDAA addr, state 4
000101	XXXXXX	XX	000110	01	10	0000	0	1	1000	00	00	01	00	00	00	LDAA addr, state 5
000110	XXXXXX	XX	000000	00	10	0000	0	1	0000	01	00	00	00	00	00	LDAA addr, state 6
000001	000101	XX	000111	01	10	0000	0	1	1000	00	00	00	00	00	00	LDAB addr, state 1
000111	XXXXXX	XX	001000	01	10	0000	0	1	1000	00	00	10	00	00	00	LDAB addr, state 7
001000	XXXXXX	XX	001001	01	10	0000	0	1	1000	00	00	01	00	00	00	LDAB addr, state 8
001001	XXXXXX	XX	000000	01	00	0000	0	1	0000	01	00	00	00	00	00	LDAB addr, state 9
000001	000110	XX	001010	01	10	0000	0	1	1000	00	00	00	00	00	00	STAA addr, state 1
001010	XXXXXX	XX	001011	01	10	0000	0	1	1000	00	00	10	00	00	00	STAA addr, state A
001011	XXXXXX	XX	001100	01	10	0000	0	1	1000	00	00	01	00	00	00	STAA addr, state B
001100	XXXXXX	XX	000000	01	10	0000	0	0	0000	01	00	00	00	00	00	STAA addr, state C
000001	000111	XX	001101	01	10	0000	0	1	1000	00	00	00	00	00	00	STAB addr, state 1
001101	XXXXXX	XX	001110	01	10	0000	0	1	1000	00	00	10	00	00	00	STAB addr, state D
001110	XXXXXX	XX	001111	01	10	0000	0	1	1000	00	00	01	00	00	00	STAB addr, state E
001111	XXXXXX	XX	000000	01	10	0001	0	0	0000	01	00	00	00	00	00	STAB addr, state F
000001	001000	XX	010000	01	10	0000	0	1	1000	00	00	00	00	00	00	LDX #data, state 1
010000	XXXXXX	XX	010001	01	10	0000	0	1	1000	00	00	00	10	00	00	LDX #data, state 10
010001	XXXXXX	XX	000000	01	10	0000	0	1	1000	00	00	00	01	00	00	LDX #data, state 11
000001	001001	XX	010010	01	10	0000	0	1	1000	00	00	00	00	00	00	LDY #data, state 1
010010	XXXXXX	XX	010011	01	10	0000	0	1	1000	00	00	00	00	10	00	LDY #data, state 12
010011	XXXXXX	XX	000000	01	10	0000	0	1	1000	00	00	00	00	01	00	LDY #data, state 13
000001	001010	XX	010100	01	10	0000	0	1	1000	00	00	00	00	00	00	LDX addr, state 1
010100	XXXXXX	XX	010101	01	10	0000	0	1	1000	00	00	10	00	00	00	LDX addr, state 14
010101	XXXXXX	XX	010110	01	10	0000	0	1	1000	00	00	01	00	00	00	LDX addr, state 15
010110	XXXXXX	XX	010111	01	10	0000	0	1	0100	01	00	00	10	00	00	LDX addr, state 16
010111	XXXXXX	XX	000000	01	10	0000	0	1	0000	01	00	00	01	00	00	LDX addr, state 17
000001	001011	XX	011000	01	10	0000	0	1	1000	00	00	00	00	00	00	LDY addr, state 1

Addr ⌀ upper 3 bytes

D5 D4 D3 D2 | D1 D0 | MSA | MSB | MSC | IRLD* | R/W | INCS | Addr Sel | PC_LD_UL* | * low true

0000 | 01 01 | 10 | 00 | 0011 | 0000 | 00 00 | Logic = 058300 Hex

LLLL LHLH | HL LL | LLLH | LLLL LL HH | Voltage = 058103 Hex

Addr 900 Hex

MSB | MSC | IRLD* | R/W | INCS | Addr sel | PC LD* | MAR LD* | X LD* | Y LD* | XDR LD | YDR LD | Logic

00 | 00 00 | 01 | 0000 | 01 | 00 | 00 00 | 00 00 | 010400 = ???? Hex

LLLL LL HH | LLLL LH HH | HH HH HH LL = 0307FC

Page Score =