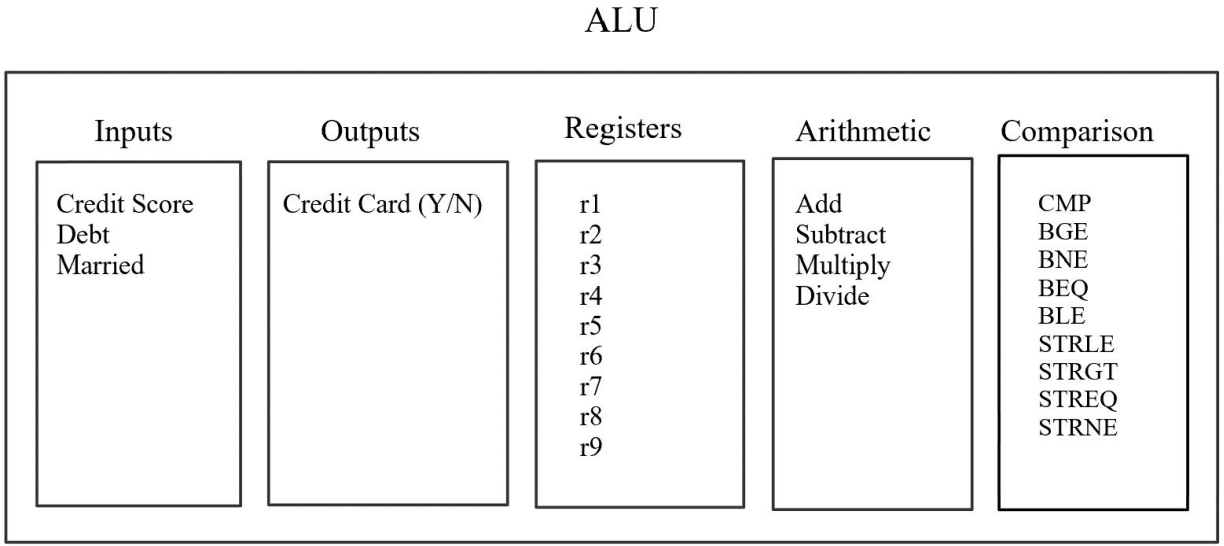
Supervised Learning in Quest (SLIQ)

By: Armis, Tyler, Jason, and Piper

Methodology

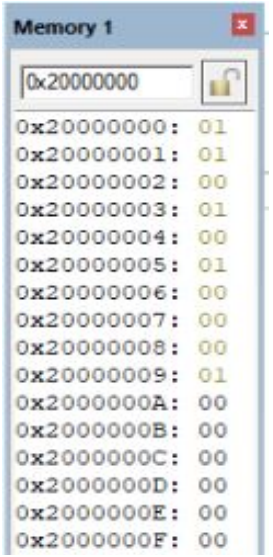
Block Diagram

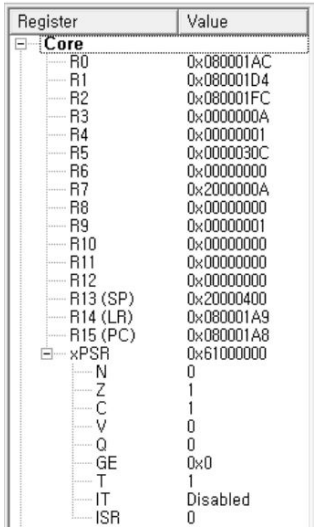


- No logic functions were used
- Kids, Income, and Age were not even used as inputs because they were on branches that were pruned

Memory

- The memory address 0x20000000 contains all of the output values
- The first three registers contain the arrays for credit score, debt, and marital status
- The arrays are stored in the memory starting at the address 0x0800019C





cscore

```
0x0800019C: 9E
0x0800019D: 02
0x0800019E: 00
0x0800019F: 00
0x080001A0: B7
0x080001A1: 02
0x080001A2: 00
0x080001A3: 00
0x080001A4: 6C
0x080001A5: 02
0x080001A6: 00
0x080001A7: 00
0x080001A8: 52
0x080001A9: 03
0x080001AA: 00
0x080001AB: 00
0x080001AC: A4
0x080001AD: 01
0x080001AE: 00
0x080001AF: 00
0x080001B0: 61
0x080001B1: 01
0x080001B2: 00
0x080001B3: 00
0x080001B4: C6
0x080001B5: 02
0x080001B6: 00
0x080001B7: 00
0x080001B8: 21
0x080001B9: 02
0x080001BA: 00
0x080001BB: 00
0x080001BC: 2E
0x080001BD: 01
0x080001BE: 00
0x080001BF: 00
0x080001C0: 0C
0x080001C1: 03
0x080001C2: 00
0x080001C3: 00
```

debt

```
0x080001C4: 00
0x080001C5: 00
0x080001C6: 00
0x080001C7: 00
0x080001C8: 00
0x080001C9: 00
0x080001CA: 00
0x080001CB: 00
0x080001CC: 14
0x080001CD: 00
0x080001CE: 00
0x080001CF: 00
0x080001D0: 00
0x080001D1: 00
0x080001D2: 00
0x080001D3: 00
0x080001D4: 03
0x080001D5: 00
0x080001D6: 00
0x080001D7: 00
0x080001D8: 00
0x080001D9: 00
0x080001DA: 00
0x080001DB: 00
0x080001DC: 1E
0x080001DD: 00
0x080001DE: 00
0x080001DF: 00
0x080001E0: 0F
0x080001E1: 00
0x080001E2: 00
0x080001E3: 00
0x080001E4: 00
0x080001E5: 00
0x080001E6: 00
0x080001E7: 00
0x080001E8: 00
0x080001E9: 00
0x080001EA: 00
0x080001EB: 00
```

married

```
0x080001EC: 00
0x080001ED: 00
0x080001EE: 00
0x080001EF: 00
0x080001F0: 01
0x080001F1: 00
0x080001F2: 00
0x080001F3: 00
0x080001F4: 00
0x080001F5: 00
0x080001F6: 00
0x080001F7: 00
0x080001F8: 01
0x080001F9: 00
0x080001FA: 00
0x080001FB: 00
0x080001FC: 01
0x080001FD: 00
0x080001FE: 00
0x080001FF: 00
0x08000200: 00
0x08000201: 00
0x08000202: 00
0x08000203: 00
0x08000204: 00
0x08000205: 00
0x08000206: 00
0x08000207: 00
0x08000208: 00
0x08000209: 00
0x0800020A: 00
0x0800020B: 00
0x0800020C: 01
0x0800020D: 00
0x0800020E: 00
0x0800020F: 00
0x08000210: 01
0x08000211: 00
0x08000212: 00
0x08000213: 00
```

Alternative and Creative Solution

- Make a seperate array for each individual person instead of array for each category. All the information for each person would have it’s own register, but we would have to be very creative seeing as there would not be enough registers for larger data sets.

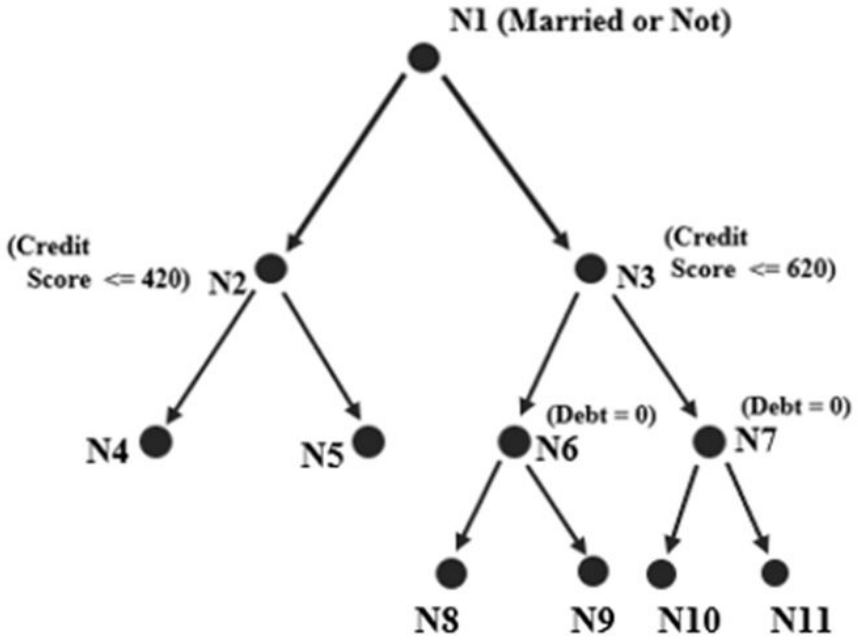
Design Limitations

- Since we used such a small data set to create the program, using very large amounts of data may cause other aspects to be more influential in the decision process than the ones we discovered to be the most profound.
- In the process of designing an adequate tree that took all the important data into account, we concluded that the person's number of kids, income, and age were not contributing factors to whether or not they received a credit card. The only factors that were considered useful were whether the person was married, had debt, and their credit score.
- For a larger data set, the code would become much more complex and some factors may become more or less important than they are in our small data set.

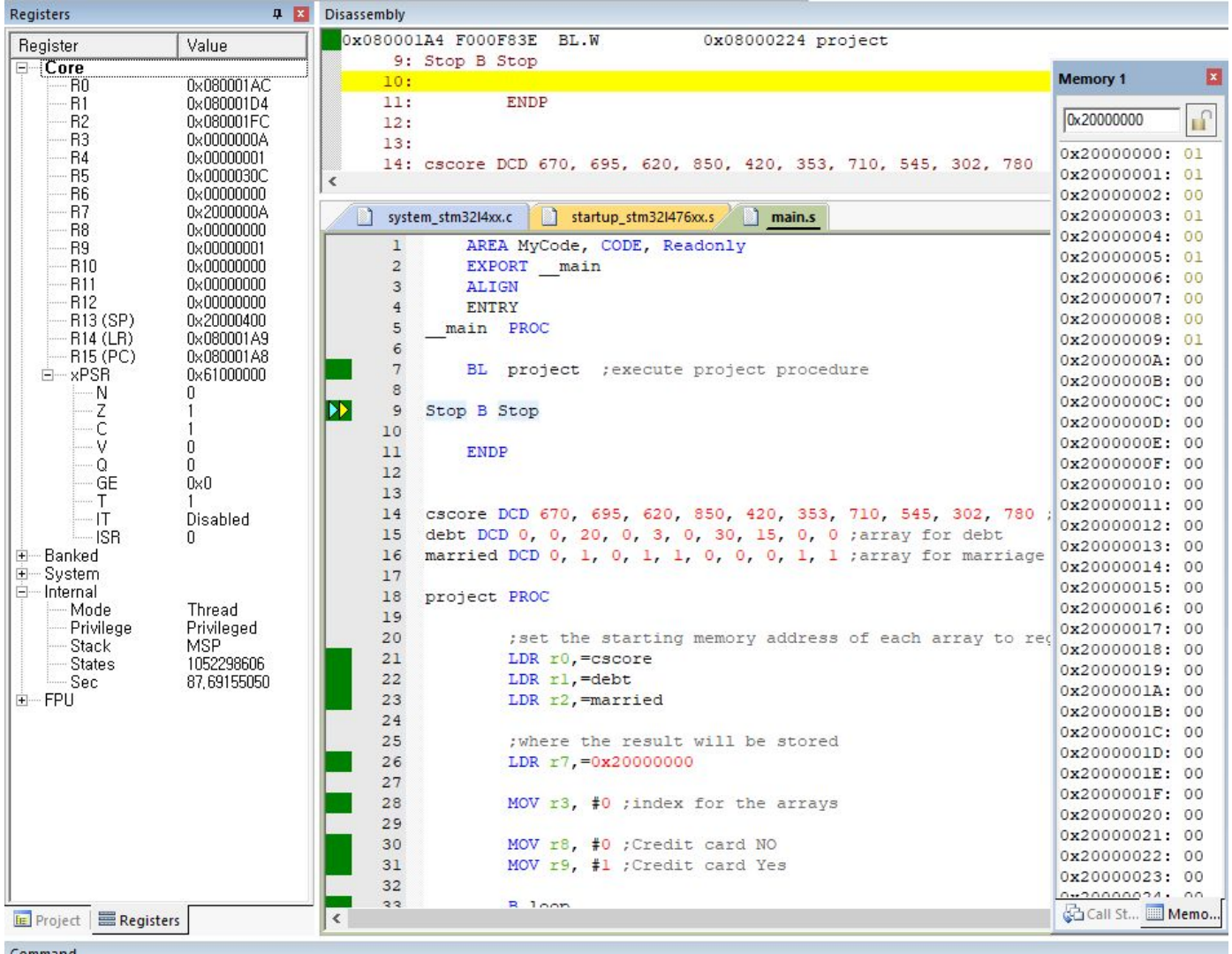
Updated Training Data

Credit Score	Debt Thousands	Married	Kids	Income Thousands	Age	Node	Credit Card?
670	0	no	0	50	18	N10	yes
695	0	yes	2	60	78	N5	yes
620	20	no	1	100	65	N9	no
850	0	yes	7	300	90	N5	yes
420	3	yes	3	43	27	N4	no
353	0	no	0	72	31	N8	yes
710	30	no	1	93	20	N11	no
545	15	no	3	45	47	N9	no
302	0	yes	4	42	58	N4	no
780	0	yes	0	37	28	N5	yes

Generated Decision Tree



```
    AREA MyCode, CODE, Readonly
    EXPORT __main
    ALIGN
    ENTRY
__main  PROC

    BL  project  ;execute project procedure

Stop B Stop

    ENDP


cscore DCD 670, 695, 620, 850, 420, 353, 710, 545, 302, 780 ;array for credit score
debt DCD 0, 0, 20, 0, 3, 0, 30, 15, 0, 0 ;array for debt
married DCD 0, 1, 0, 1, 1, 0, 0, 0, 1, 1 ;array for marriage status

project PROC

        ;set the starting memory address of each array to registers
        LDR r0,=cscore
        LDR r1,=debt
        LDR r2,=married

        ;where the result will be stored
        LDR r7,=0x20000000

        MOV r3, #0 ;index for the arrays

        MOV r8, #0 ;Credit card NO
        MOV r9, #1 ;Credit card Yes

        B loop

loop    CMP r3, #10 ;while index < 10
        BGE endloop

        ;set each array[index] values to registers
        LDR r4, [r2, r3, LSL #2] ;married
        LDR r5, [r0, r3, LSL #2] ;cscore
        LDR r6, [r1, r3, LSL #2] ;debt

        ;first split
        CMP r4, #1 ;compare marriage status with 1 then split
        BEQ loopTyler
        BNE loopJason

        ;split on N2
loopTyler CMP r5, #420 ;compare credit score with 420 then split
        BLE loop1
        BGT loop2

loop1   STRB r8, [r7],#1 ;node(N4) is no
        ADD r3, r3, #1 ;index++
        B loop ;back to loop

loop2   STRB r9, [r7],#1 ;node(N5) is yes
        ADD r3, r3, #1 ;index++
        B loop ;back to the loop

        ;split on N3
loopJason CMP r5, #620 ;compare credit score with 620 then split

        BLE loopArmis

        ;if it is greater than 620 (split on N7)
        CBZ r6, loop3 ;compare and branch if zero
        CBNZ r6, loop4 ;compare and branch if not zero

loop3   STRB r9, [r7],#1 ;node(N10) is yes
        ADD r3, r3, #1 ;index++
        B loop ;back to the loop

loop4   STRB r8, [r7],#1 ;node(N11) is no
        ADD r3, r3, #1 ;index++
        B loop ;back to the loop

        ;split on N6
loopArmis
        CBZ r6, loop5 ;compare and branch if zero
        CBNZ r6, loop6 ;compare and branch if not zero

loop5   STRB r9, [r7],#1 ;node(N8) is yes
        ADD r3, r3, #1 ;index++
        B loop ;back to the loop

loop6   STRB r8, [r7],#1 ;node(N9) is no
        ADD r3, r3, #1 ;index++
        B loop ;back to the loop

endloop
        BX LR ;return
    ENDP
    END
```


Registers
Register
Value
Core
R0 0x080001AC
R1 0x080001D4
R2 0x080001FC
R3 0x0000000A
R4 0x00000001
R5 0x0000030C
R6 0x00000000
R7 0x2000000A
R8 0x00000000
R9 0x00000001
R10 0x00000000
R11 0x00000000
R12 0x00000000
R13 (SP) 0x20000400
R14 (LR) 0x080001A9
R15 (PC) 0x080001A8
xPSR 0x61000000
N 0
Z 1
C 1
V 0
Q 0
GE 0x0
T 1
IT Disabled
ISR 0
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 1052298606
Sec 87.69155050
FPU
Project
Registers
Disassembly
0x080001A4 F000F83E BL.W 0x08000224 project
9: Stop B Stop
10:
11: ENDP
12:
13:
14: cscore DCD 670, 695, 620, 850, 420, 353, 710, 545, 302, 780
system_stm32l4xx.c
startup_stm32l476xx.s
main.s
1 AREA MyCode, CODE, Readonly
2 EXPORT __main
3 ALIGN
4 ENTRY
5 __main PROC
6
7 BL project ;execute project procedure
8
9 Stop B Stop
10
11 ENDP
12
13
14 cscore DCD 670, 695, 620, 850, 420, 353, 710, 545, 302, 780 ;
15 debt DCD 0, 0, 20, 0, 3, 0, 30, 15, 0, 0 ;array for debt
16 married DCD 0, 1, 0, 1, 1, 0, 0, 0, 1, 1 ;array for marriage
17
18 project PROC
19
20 ;set the starting memory address of each array to reg
21 LDR r0,=cscore
22 LDR r1,=debt
23 LDR r2,=married
24
25 ;where the result will be stored
26 LDR r7,=0x20000000
27
28 MOV r3, #0 ;index for the arrays
29
30 MOV r8, #0 ;Credit card NO
31 MOV r9, #1 ;Credit card Yes
32
Memory 1
0x20000000
0x20000000: 01
0x20000001: 01
0x20000002: 00
0x20000003: 01
0x20000004: 00
0x20000005: 01
0x20000006: 00
0x20000007: 00
0x20000008: 00
0x20000009: 01
0x2000000A: 00
0x2000000B: 00
0x2000000C: 00
0x2000000D: 00
0x2000000E: 00
0x2000000F: 00
0x20000010: 00
0x20000011: 00
0x20000012: 00
0x20000013: 00
0x20000014: 00
0x20000015: 00
0x20000016: 00
0x20000017: 00
0x20000018: 00
0x20000019: 00
0x2000001A: 00
0x2000001B: 00
0x2000001C: 00
0x2000001D: 00
0x2000001E: 00
0x2000001F: 00
0x20000020: 00
0x20000021: 00
0x20000022: 00
0x20000023: 00
Call St...
Memo...