FSU panama city

EEL 4746 Microprocessor System Design

Performed by:

Tyler Coltrane, Piper Ellsworth, Armis Sunday, and Jason Lee

Instructor: Dr. Manzak

April 21st, 2022

Sequence Learning in Quest

Abstract

SLIQ (Supervised Learning in Quest) is a high speed and flexible decision tree classifier developed by IBM Almaden Research Center in 1996 that allows us to sort and interpret data. Data classification is a bottleneck in data mining due to other methods' inability to scale with large data sets spread across different classifications. SLIQ can reduce costs using efficient and pre-sorting decision trees to sort through larger data sets while accounting for differences in data types. This maintains competitive accuracy with the ability to scale and interpret larger data sets with multiple classes and attributes.

Introduction

SLIQ is a decision tree algorithm. This means it splits nodes into two or more sub-nodes based on some criteria. As more sub-nodes are created the bucketed data’s homogeneity increases, the data becomes more similar, and the purity of the node increases with respect to the target variable. SLIQ improves on this framework by aiming to reduce the diversity of the tree at each split. This allows SLIQ to sort through data more efficiently and cost-effectively over large data sets with different data types[1].

SLIQ uses a training set and a Gini split to prepare the data for the decision tree algorithm. These equations are what make SLIQ a,” supervised learning,” algorithms as the data is pre-sorted and pruned. For training set L with n distinct classes the equation is

$$Gini(L) = 1 - \sum j = 1 \ldots n \; P^2 j \qquad (1)$$

- pj is the relative frequency of j.

After the binary split of the set L into sets L1 and L2 the index becomes,

$$Ginisplit(L) = \frac{|L1|}{|L|} Gini(L1) + \frac{|L2|}{|L|} Gini(L2). \qquad (2)$$

Lastly, the Gini Index equation is with data classes a and b is

$$Gini\ Index = \frac{a1 + a2}{n}\left[1 - \left(\frac{a1}{a1 + a2}\right)^2 - \left(\frac{a2}{a1 + a2}\right)^2\right] +$$
$$\frac{b1 + b2}{n}\left[1 - \left(\frac{b1}{b1 + b2}\right)^2 - \left(\frac{b2}{b1 + b2}\right)^2\right] (5)$$

(3). These mathematical formulations are what sets SLIQ algorithms apart from traditional decision tree algorithms.

SLIQ is an improvement upon existing decision tree algorithms. SLIQ’s advantages are based in its pre-sorting of data, no need for data normalization, scales well with data size, and it can handle a variety of data types across many features and classifications. The downsides to SLIQ lie in its complexity, time, and cost. SLIQ can very quickly become very complex for large data sets, requires more time to train the model, is more expensive and complex than a normal decision tree algorithm, and cannot be applied to regressions or predictive modeling [2].

The application of SLIQ can be used in any field where data mining is prevalent. Specifically, SLIQ is being used today in the deregulated power market. The SLIQ algorithm allows us to mine data in terms of cost of energy for purchase and sale to meet load demands and decrease the cost of energy usage across any industry [4].

Existing Work

Throughout the research of SLIQ, researchers found that there is a lack of studies about the algorithm but were unable to find it’s first inventors. From the IEEE Xplore library, researchers were able to find a paper from 1996 written by three engineers: Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. These engineers proposed that SLIQ could solve an important problem at the time, data mining. The algorithm for SLIQ creates a decision tree that can handle both numeric and categorial attributes, which uses presorting techniques and optimizations to create the ideal results[4]. These engineers suggested that one could use the algorithm to create inexpensive, compact, and accurate trees.

Within the first research paper reveiwed, the researchers reviewed previous studies conducted on classification, but found that for large data sets, they don't scale well. As a solution, they proposed a decision- tree classifier, SLIQ, designed specifically for scalability.

Within the second research paper reveiewed, researchers found that the prediction with the greatest separating power correlates to a split in a decision tree. The optimal split creates nodes where a single class dominates the most[5]. The predictor's power to separate data may be calculated in a variety of ways. The Gini coefficient of inequality is one of the most well-known methodologies.

Within the second research paper reveiewed, the researchers used data mining for an easy tool to analyze historical rainfall data, it allowed them to measure valuable patterns within a short period of time. With an average accuracy of 74.92 percent, the SLIQ decision tree algorithm was able to estimate an accurate precipitation forecast[6].

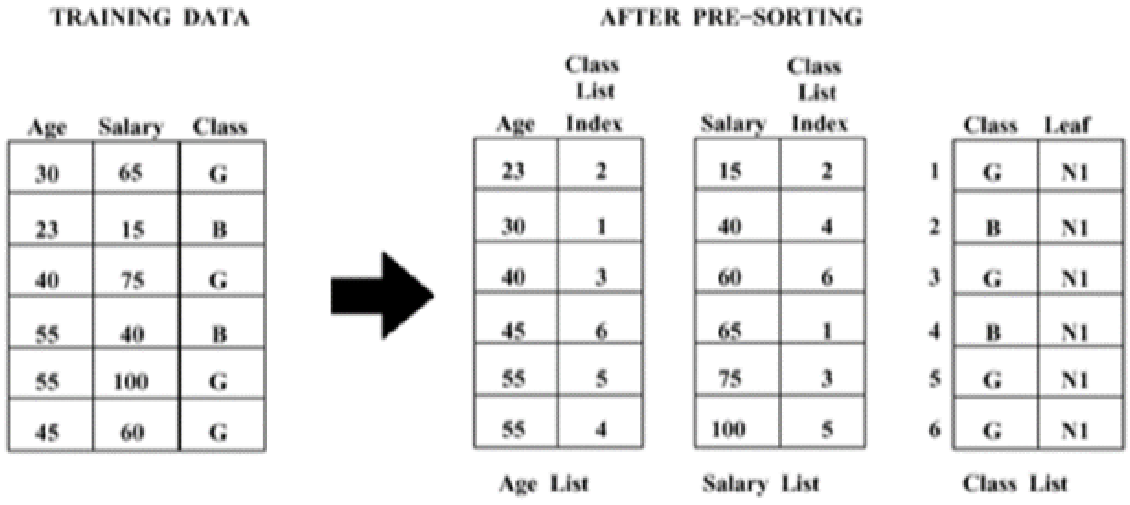
In 2005, through the International Power Engineering Conference, three engineers, named Hongwen Yan, Rui Ma, and Xiaojiao Tong proposed using SLIQ to build a framework for a competitive bidding assessment in a deregulated power market. They suggested that the bidding system using the SLIQ algorithm could be consistent with the features of the electric energy production and consummation, this would be more convenient for operating the power markets[7].

Throughout the research of this algorithm, researchers only found software implementations. Where SLIQ is used to make decisions based on a set of data for a specific use. Of these software implementations, there are examples algorithms of precipitation prediction, bidding, or sorting data. There are also algorithms that identifies households that are most likely to respond to a promotion of a product, such as a new banking service[5]. Researchers have found that the SLIQ decision tree, superior to other algorithms, can be built fast and scalable for larger data sets.

Example

- Algorithm

SLIQ is a decision tree classifier that can improve learning time for the classifier without any loss in accuracy while this technique allows performing on the larger training data. SLIQ uses Gini Index to determine the best split for each node [8]. SLIQ Algorithm can be divided into 3 steps, pre-sorting the sample, processing evaluation on splits, and updating the class list [4].



TRAINING DATA

Age	Salary	Class
30	65	G
23	15	B
40	75	G
55	40	B
55	100	G
45	60	G

AFTER PRE-SORTING

Age List

Age	Class List Index
23	2
30	1
40	3
45	6
55	5
55	4

Salary List

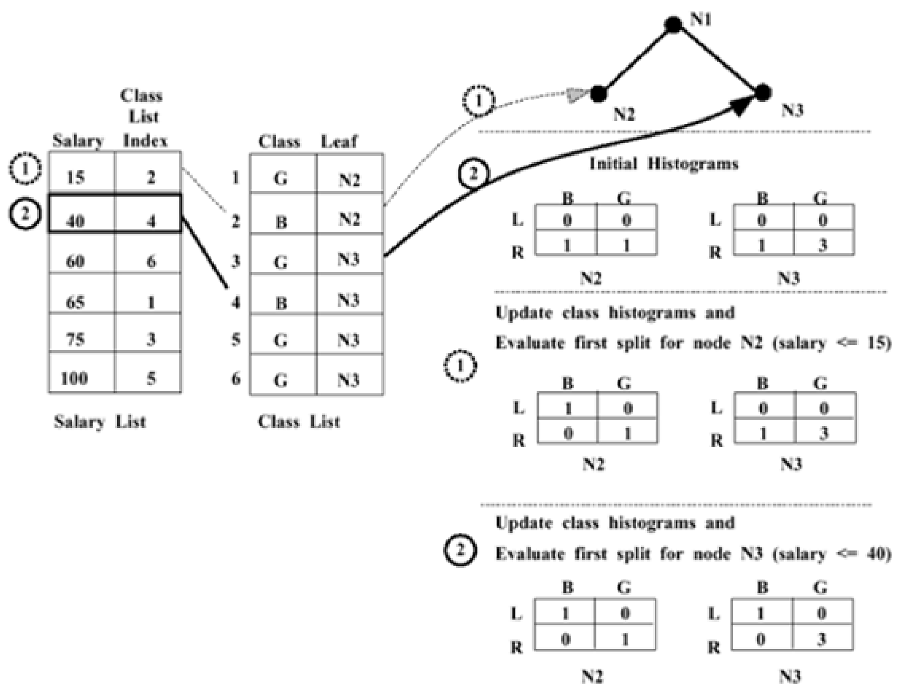
Salary	Class List Index
15	2
40	4
60	6
65	1
75	3
100	5

Class List

	Class	Leaf
1	G	N1
2	B	N1
3	G	N1
4	B	N1
5	G	N1
6	G	N1

Pre-sorting the sample with given training data, create an attribute list for each attribute.

Once the data is sorted, it can process evaluation on splits by assuming the split of the first node and evaluate each histogram. (Gini index will be used during this step but for this example since attribute, age and salary do not have predictive power since they can be any number.)


N1
N2
N3
1
2
Initial Histograms
Update class histograms and
Evaluate first split for node N2 (salary <= 15)
Update class histograms and
Evaluate first split for node N3 (salary <= 40)

Salary List

Salary	Class List Index
15	2
40	4
60	6
65	1
75	3
100	5

Class List

	Class	Leaf
1	G	N2
2	B	N2
3	G	N3
4	B	N3
5	G	N3
6	G	N3

Initial Histograms

N2	B	G
L	0	0
R	1	1

N3	B	G
L	0	0
R	1	3

Update class histograms and Evaluate first split for node N2 (salary <= 15)

N2	B	G
L	1	0
R	0	1

N3	B	G
L	0	0
R	1	3

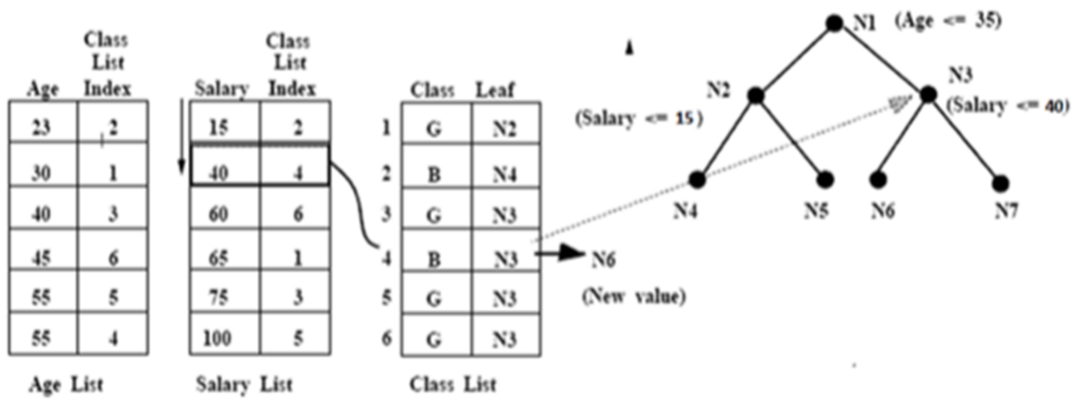
Update class histograms and Evaluate first split for node N3 (salary <= 40)

N2	B	G
L	1	0
R	0	1

N3	B	G
L	1	0
R	0	3

- L indicates branch left.
- R indicates branch right.

Finally the class list can be updated. Traverse the training data through the decision tree and replace the node with the new node. These steps can be repeated until each of the leaf nodes becomes a pure node, meaning that the node only contains one class.



Age List

Age	Class List Index
23	2
30	1
40	3
45	6
55	5
55	4

Salary List

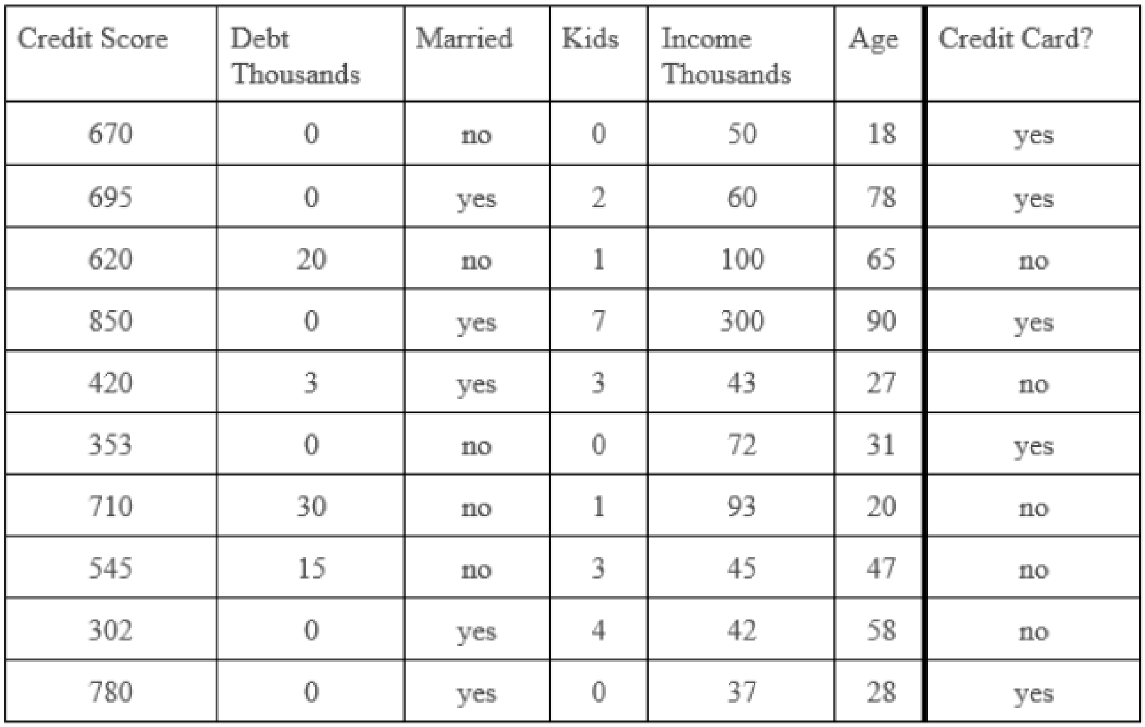
Salary	Class List Index
15	2
40	4
60	6
65	1
75	3
100	5

Class List

	Class	Leaf
1	G	N2
2	B	N4
3	G	N3
4	B	N3
5	G	N3
6	G	N3

- Example
    - Question

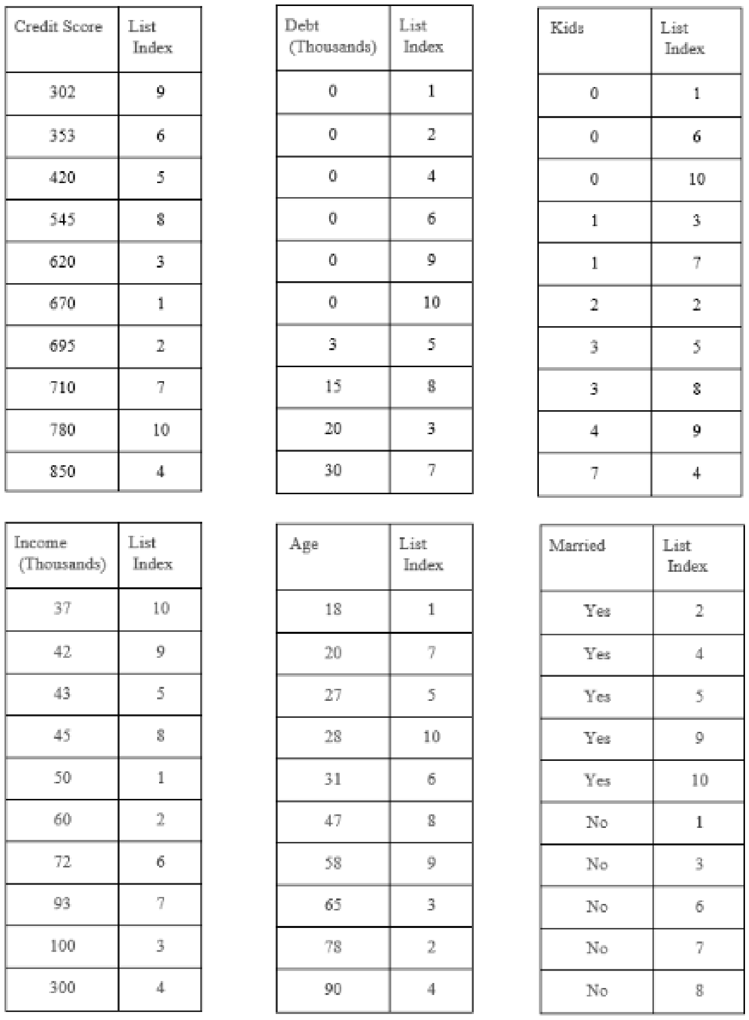
The question for the example given is whether an individual will be chosen to receive a credit card. The decision tree shows the results using the given data.



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

Example Training Data

- Solution



Credit Score	List Index
302	9
353	6
420	5
545	8
620	3
670	1
695	2
710	7
780	10
850	4

Debt (Thousands)	List Index
0	1
0	2
0	4
0	6
0	9
0	10
3	5
15	8
20	3
30	7

Kids	List Index
0	1
0	6
0	10
1	3
1	7
2	2
3	5
3	8
4	9
7	4

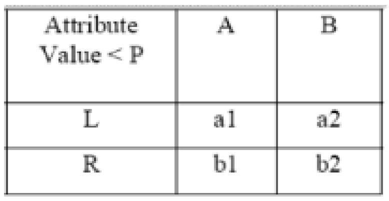
Income (Thousands)	List Index
37	10
42	9
43	5
45	8
50	1
60	2
72	6
93	7
100	3
300	4

Age	List Index
18	1
20	7
27	5
28	10
31	6
47	8
58	9
65	3
78	2
90	4

Married	List Index
Yes	2
Yes	4
Yes	5
Yes	9
Yes	10
No	1
No	3
No	6
No	7
No	8

Pre-sorted data with index list

Determine the Gini index using the equation and the histogram made based upon the data of the attributes. Where P is the value of the attribute obtained from the sorted data, A is true, B is fails, L is branch left, R is branch right, and a and b is the number of the elements that fall into the criteria of the table.



Attribute Value < P	A	B
L	a1	a2
R	b1	b2

Histogram example

$$Gini\ Index = \frac{a1+a2}{n}\left[1-\left(\frac{a1}{a1+a2}\right)^2-\left(\frac{a2}{a1+a2}\right)^2\right]+\ \frac{b1+b2}{n}\left[1-\left(\frac{b1}{b1+b2}\right)^2-\left(\frac{b2}{b1+b2}\right)^2\right]$$

Gini index equation

Using the equation, Gini index for the overall training data is $1 – (5/10)^2 - (5/10)^2 = 0.5$.

Gini index for all the attributes that are sorted are 0 because their new data cannot be predict, meaning that they have to predictive power.

Gini index for the marriage status attribute is

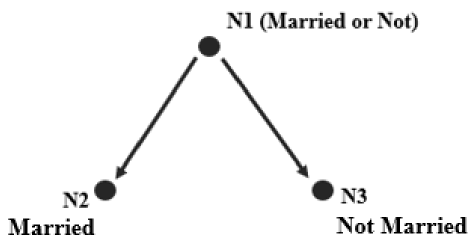
$$0.5\ [1 – (3/5)^2 – (2/5)^2] + 0.5\ [1 – (2/5)^2 – (3/5)^2] = 0.48$$

Since the table created by within the marriage status attribute is:

Married?	A	B
L	3	2
B	2	3

From the Gini indexes found, all the other attributes have zero Gini index but marriage status, the marriage status attribute must be used to split the data from the first node.

Giving a decision tree and two other histograms as:



N2 = Married	Yes	No
L	0	0
R	3	2

N3 = Not Married	Yes	No
L	0	0
R	2	3

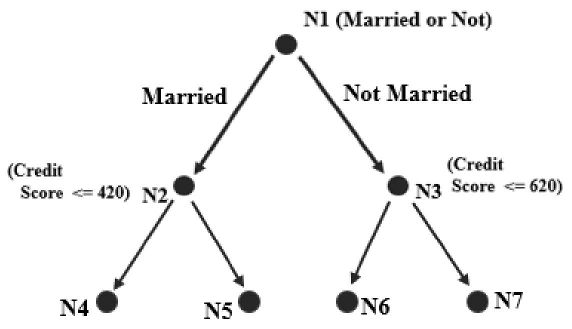
From the N2 and N3, any of the attribute can be take in place to split the node since they all had the zero Gini index. Since it is obvious that the credit score matters the most towards the question, use the credit score attribute to split the N2 and N3.

N2 (<= 420)	Yes	No
L	0	2
R	3	0

N3 (<= 620)	Yes	No
L	1	2
R	1	1

The N2 has been split nicely into N4 and N5 but the N3 has not. So repeat the step with different attribute on N6.

The steps needs to be repeated until each of the leaf node becomes a pure node, meaning that the node only contains one result.



On the Node 6 and Node 7 debt in thousands attribute can be used for splitting the results.

N6	Yes	No
L	1	2
R	0	0

$\longrightarrow$

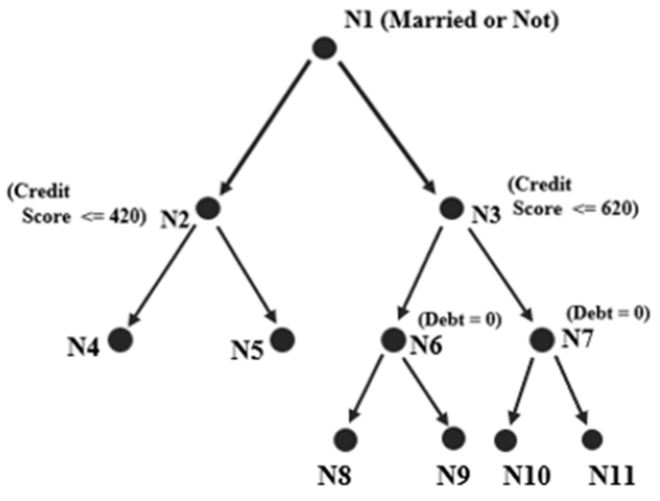
N6 (= 0)	Yes	No
L	1	0
R	0	2

N7	Yes	No
L	0	0
R	1	1

$\longrightarrow$

N7 (= 0)	Yes	No
L	1	0
R	0	1

With the finalized decision tree update the training data by running the data through the decision tree and replace the node with rather the credit card can be issued or not.



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

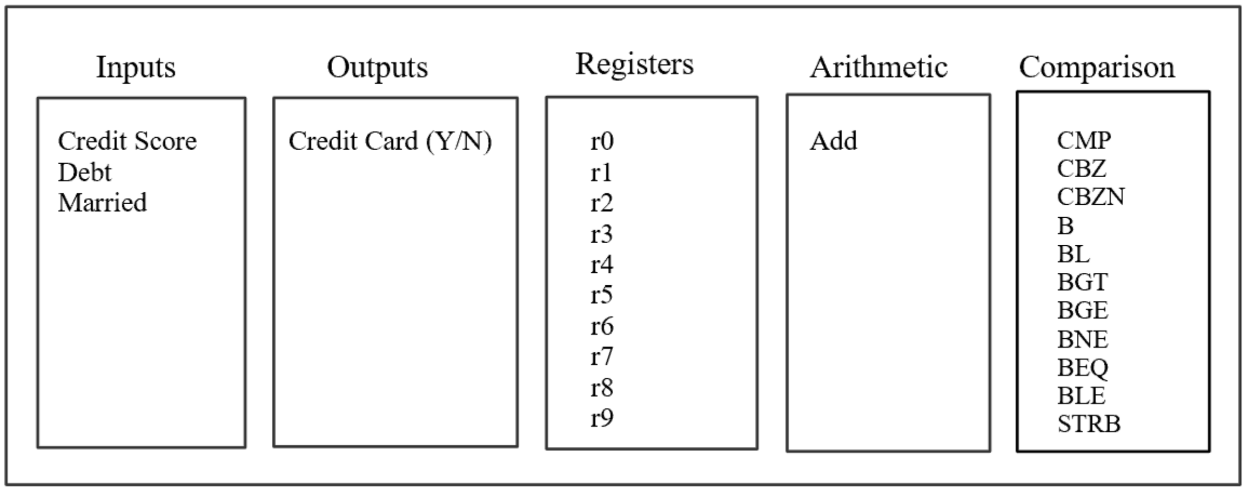
Updated training data

Methodology

As seen in the previous example, four students created a program in order to determine whether or not a person is eligible to receive a credit card based on a set of data. The factors include the person's current credit score, debt, marital status, number of children, income, and age.

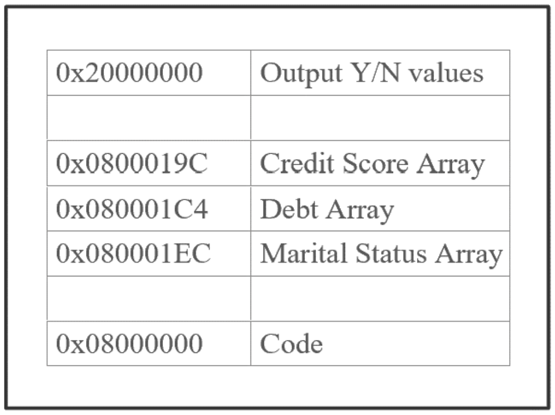
Below are two block diagrams one labeled ALU and the other Memory. The ALU diagram was derived from the code which was written by the team of students. Within the ALU, the first box is the inputs. The only inputs that were used in the program were the person's credit score, debt, and marital status. As the students dove into the test data, they discovered some of the information in the table did not have an impact on whether a person received a credit card. Since these other factors were not significant, they were pruned off of the decision tree which was used to write the code. Due to them being pruned, there are only three inputs into the program. The insignificant factors include number of children, income, and age. The second box in the ALU labeled outputs includes only one output. The only thing the program is outputting is a yes or no of whether the person should get a credit card or not. Yes is a 01, and no is a 00. The output can be seen by accessing the memory. The next box inside of the ALU contains the registers. The students used 10 registers in their code. The first three registers r0, r1, and r2 were used to hold the credit score, debt, and married arrays. Register 3 was used to keep track of the index for each array. Registers 4, 5, and 6 were used to hold specific elements pulled out of the arrays. R7 was used to store the output at the memory location 0x20000000. Registers 8 and 9 were set as 0 and 1 to represent no and yes as outputs in the memory. In the section of the ALU titles arithmetic there is only the addition function. The ADD function was used to update the index in register 3. No other arithmetic functions were used. The last ALU box holds all the comparisons from the code. The code contains 11 different comparison functions. Many of them appear multiple times in the code. They proved to be very useful for SLIQ and its decision tree algorithm. Other types of functions performed in the ALU include logic functions. The block diagram does not contain any logic functions because the students did not use any.

ALU



The figure below is the block diagram of the memory. The memory address 0x20000000 contains all of the output values in the form of ones and zeros. The array for the credit score is stored in the memory starting at 0x0800019C. Starting at the memory address 0x080001C4 the array is holding the data containing the people's debt. The marital status array is being stored at the address 0x08001EC. The rest of the code is being stored at the memory address 0x08000000.

Memory



0x20000000	Output Y/N values
0x0800019C	Credit Score Array
0x080001C4	Debt Array
0x080001EC	Marital Status Array
0x08000000	Code

Though the way the code was written was great, there are some other ways it could have been written. One thing the students could have done differently was implement the Gini index into the code. One of the students calculated the Gini index by hand to create the decision tree that the code was based off of. As an alternative, the students could have incorporated the Gini index into the program along with all of the data from the table and let the program decide what was more important. However there are issues with this option. To start, the current program is already using 10 registers, and just by including the three other arrays as inputs would require 13 registers. Then they would probably need another two or three registers just to calculate the Gini index. The total number of registers for this method would be about 16, and assembly does not even have that many available registers. In addition to that, the program would probably end up being at least triple the size of the current program making it extremely unlikely for the students to be able to complete in the allotted time. Another alternative solution to the current program would be to create an array for each person as opposed to each category. This would allow less loops to have to be used within the code which would ultimately shorten the code and make it easier to read, but there are quite a few issues with this route. For this particular example, there are ten people which would occupy ten registers as opposed to three before. For a larger data set, even more registers would need to be occupied and assembly just does not have enough to be able to do that.

This design by the students does have some limitations however. Since they 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 they discovered to be. The larger data set may cause some of the pruned branches to become more important and useful. For a larger data set, the code would become much more complex than it is right now. Depending on how large the data set becomes, it may be easier to figure out how to add the calculation of the Gini index into the code, so that it does not have to be performed by hand.

While the students were actually working on the project, Piper did the full methodology while Jason, Armis, and Tyler wrote the code. For the most part they all wrote the code together, but there were some portions where they worked individually. The code is full of loops. The loop called “loopTyler” is the part Tyler coded on his own. The loop called “loopJason” is the part Jason did on his own, and the loop called “loopArmis” is the part Armis did on his own.

Results

Within the assembly code of the SLIQ algorithm, students found it was easier to assign each person applying for the credit card a specific number. This specific number would allign to their data in the different arrays for the inputs, as seen in the assembly code. With the same manner, the final answer of yes or no for the person applying will be provided as an array located in the memory. Decoding the array for the results, a “01” would clarify that yes this person can have the credit card as for a “00” would clarify that no this person can not have a credit card. Within the assembly code, the solution could easily be scaled larger for more data to be input and more precise decisions. Alternatively, their code could have been done with the opposite branch comparisons. Where a BLT is, it could be replaced with a BGE and have the code aligned to match the procedure of decision tree. Other solutions would be longer and more complex without the compare branch functions. Students took this complexity in consideration and decided that using these branches to our advantage would have the best results.

Conclusion

In this project, students successfully used the supervised learning in quest (SLIQ) algorithm to determine whether or not a person should receive a credit card. They used a data set and Gini to determine which information was most important and which was not. They were then able to create a decision tree with multiple branches. Afterwards, they used the decision tree to serve as a guide to the coding process. Like how a decision tree makes many comparisons, the program includes many different comparison functions. From this project, the students learned the history behind SLIQ and decision tree algorithms, they also used team work and improved their troubleshooting skills when things went wrong in the coding process. For future work, the students would alter their code so that it could handle larger sets of data like SLIQ is designed for. They would also like to implement their design onto hardware where the board could receive inputs and give back outputs.

References

[1] Huacheng Zhang, Wu Xie, "Improvement of SLIQ Algorithm and its Application in Evaluation", Genetic and Evolutionary Computing 2009. WGEC '09. 3rd International Conference on, pp. 77-80, 2009.

[2] Xie Wu, Huacheng Zhang, Huimin Zhang, "Study of comprehensive evaluation method of undergraduates based on data mining", Intelligent Computing and Integrated Systems (ICISS) 2010 International Conference on, pp. 541-543, 2010.

[3] Hongwen Yan, Rui Ma and Xiaojiao Tong, "SLIQ in data mining and application in the generation unit's bidding decision system of electricity market," 2005 International Power Engineering

[4] Mehta, M., Agrawal, R., Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In: Apers, P., Bouzeghoub, M., Gardarin, G. (eds) Advances in Database Technology — EDBT '96. EDBT 1996. Lecture Notes in Computer Science, vol 1057. Springer, Berlin, Heidelberg.

[5] S. Sivagama Sundhari, "A knowledge discovery using decision tree by Gini coefficient," 2011 International Conference on Business, Engineering and Industrial Applications, 2011, pp. 232-235.

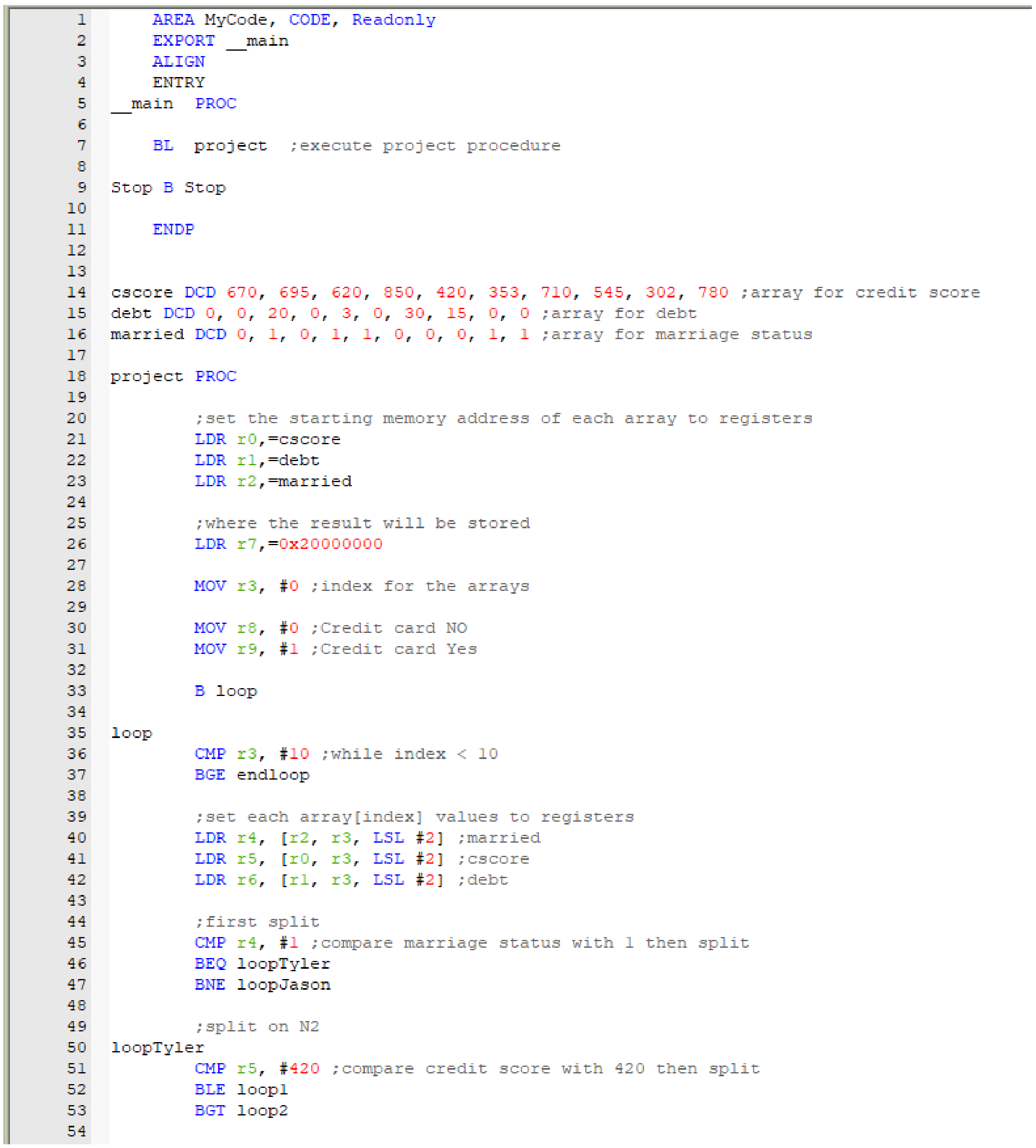
[6] N. Prasad, P. K. Reddy and M. M. Naidu, "A Novel Decision Tree Approach for the Prediction of Precipitation Using Entropy in SLIQ," 2013 UKSim 15th International Conference on Computer Modelling and Simulation, 2013, pp. 209-217.

[7] Hongwen Yan, Rui Ma and Xiaojiao Tong, "SLIQ in data mining and application in the generation unit's bidding decision system of electricity market," 2005 International Power Engineering Conference, 2005, pp. 1-137.

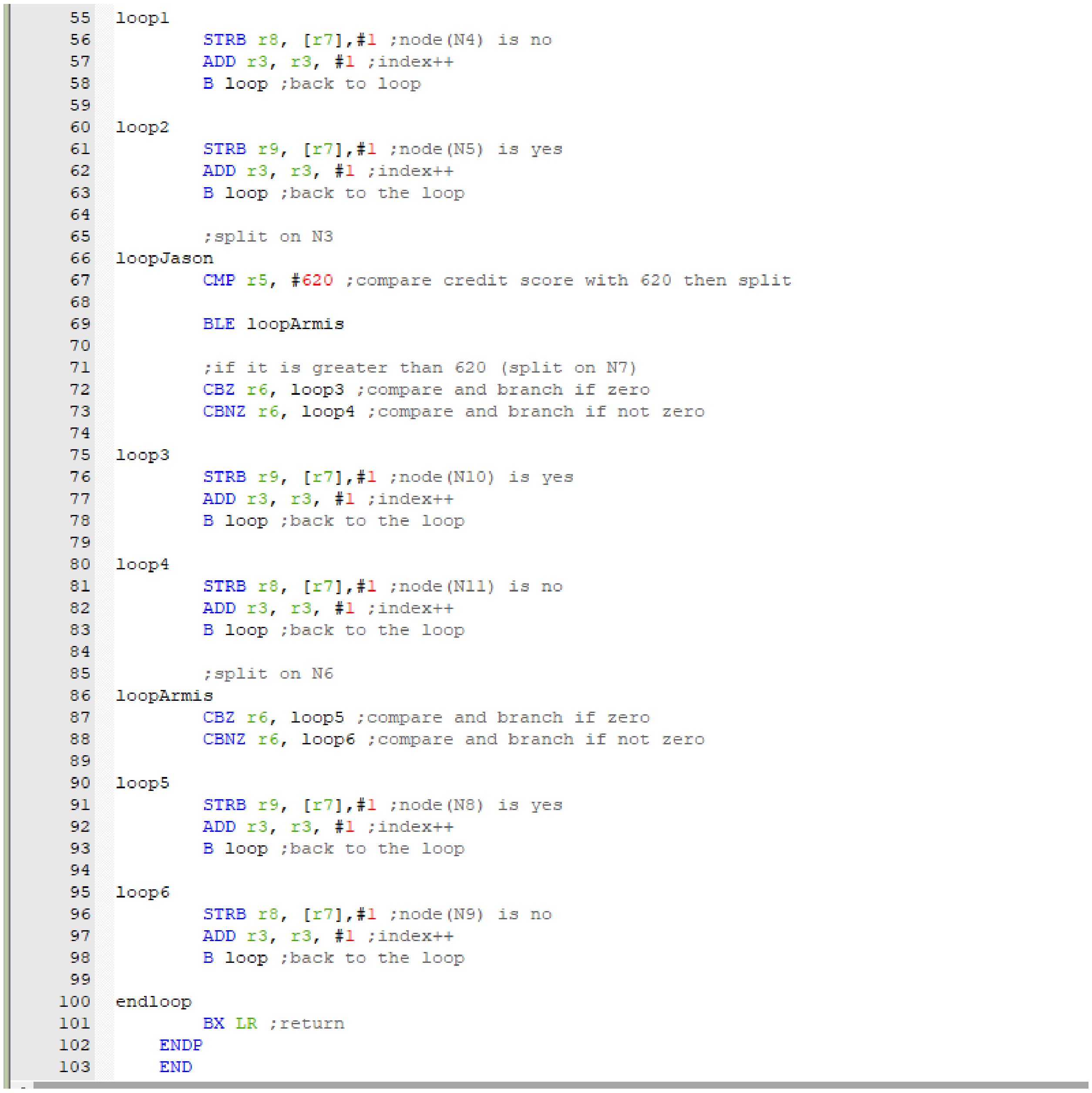
[8] B. Chandra and V. P. Paul, "A Robust Algorithm for Classification Using Decision Trees," 2006 IEEE Conference on Cybernetics and Intelligent Systems, 2006, pp. 1-5

Appendix

Assembly Code



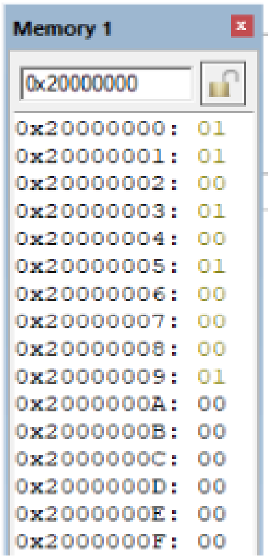
```
    AREA MyCode, CODE, Readonly
    EXPORT __main
    ALIGN
    ENTRY
__main  PROC

    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
```

Simulation Results

- Memory

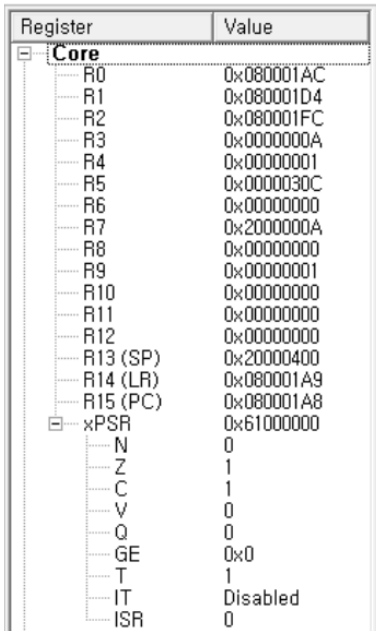


Memory 1

0x20000000

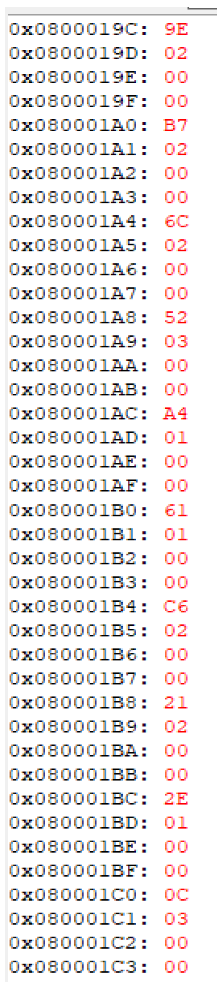
```
0x20000000: 01
0x20000001: 01
0x20000002: 00
0x20000003: 01
0x20000004: 00
0x20000005: 01
0x20000006: 00
0x20000007: 00
0x20000008: 00
0x20000009: 01
0x2000000A: 00
0x2000000B: 00
0x2000000C: 00
0x2000000D: 00
0x2000000E: 00
0x2000000F: 00
```

- Register



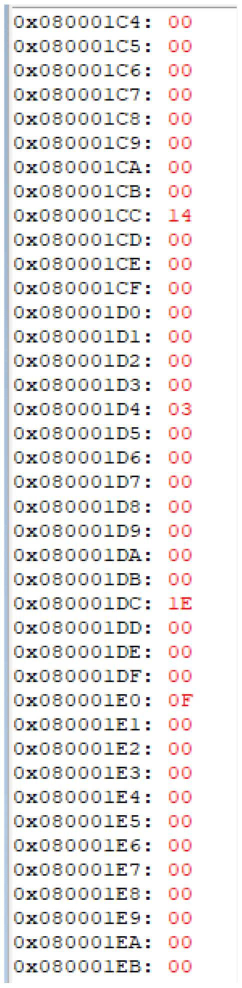
```
Register          Value
Core
  R0              0x080001AC
  R1              0x080001D4
  R2              0x080001FC
  R3              0x0000000A
  R4              0x00000001
  R5              0x0000030C
  R6              0x00000000
  R7              0x2000000A
  R8              0x00000000
  R9              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
```

- Data memories
    - Credit score



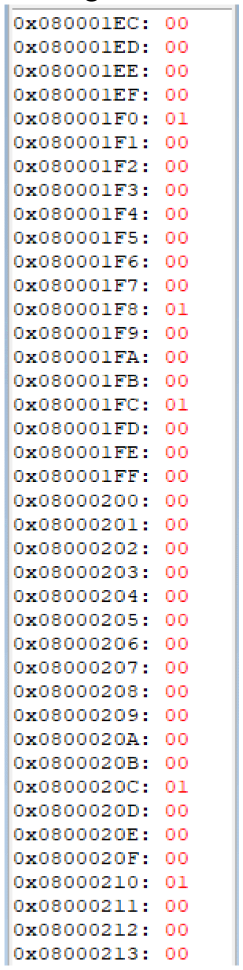
```
0x0800019C: 9E
0x0800019D: 02
0x0800019E: 00
0x0800019F: 00
0x080001A0: B7
0x080001A1: 02
0x080001A2: 00
0x080001A3: 00
0x080001A4: 6C
0x080001A5: 02
0x080001A6: 00
0x080001A7: 00
0x080001A8: 52
0x080001A9: 03
0x080001AA: 00
0x080001AB: 00
0x080001AC: A4
0x080001AD: 01
0x080001AE: 00
0x080001AF: 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



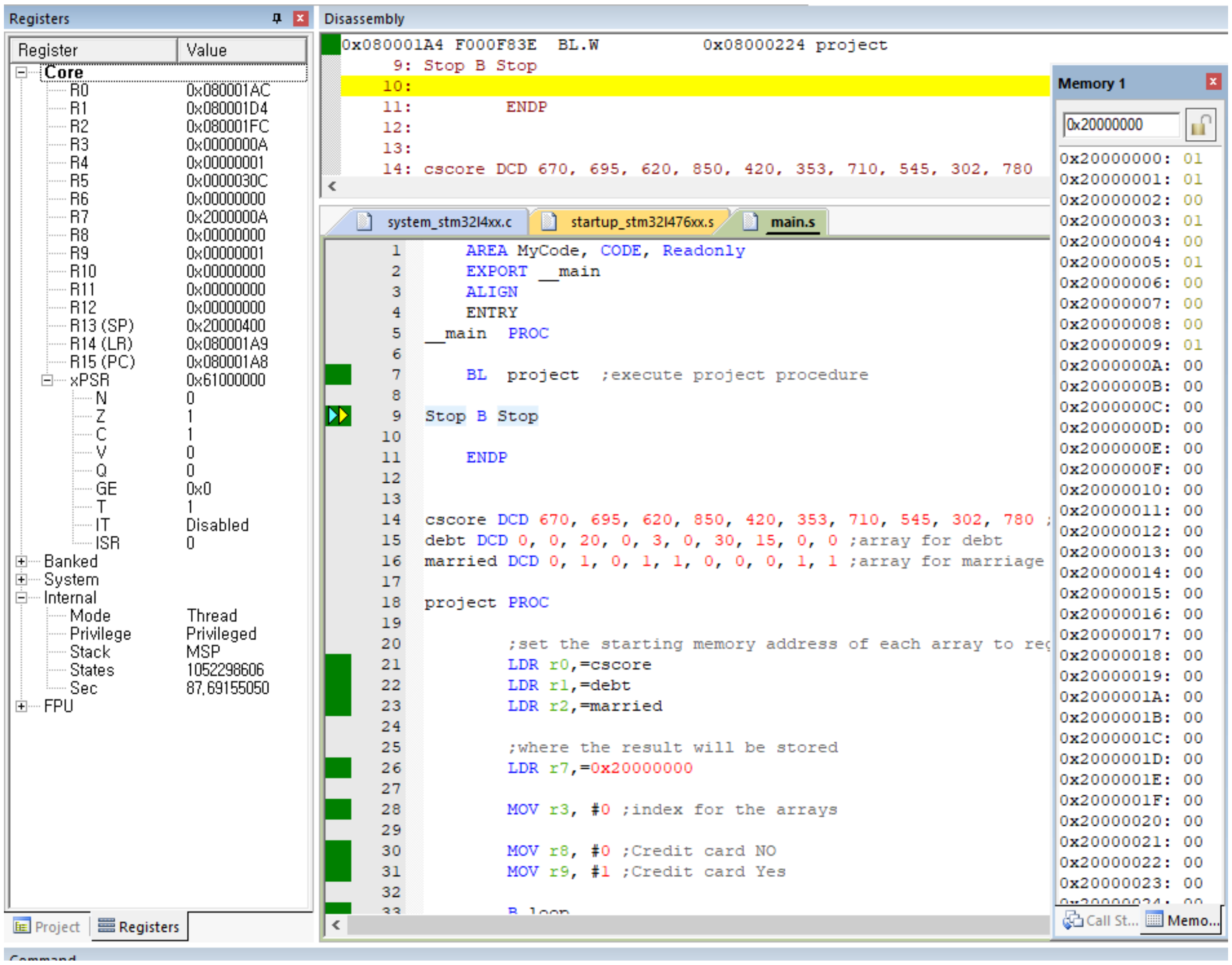
```
0x080001C4: 00
0x080001C5: 00
0x080001C6: 00
0x080001C7: 00
0x080001C8: 00
0x080001C9: 00
0x080001CA: 00
0x080001CB: 00
0x080001CC: 14
0x080001CD: 00
0x080001CE: 00
0x080001CF: 00
0x080001D0: 00
0x080001D1: 00
0x080001D2: 00
0x080001D3: 00
0x080001D4: 03
0x080001D5: 00
0x080001D6: 00
0x080001D7: 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
```

    - Marriage status



```
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
```

- Full view



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
```

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

```
    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 ;
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

project PROC

        ;set the starting memory address of each array to reg
        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
```

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
```