Istanbul Technical University
Department of Computer Engineering

BLG 212E – Microprocessor Systems
2020-2021 Term Project

Due Date: 31.01.2021, **Sunday,** 23:59.

You are expected to create a **sorted set linked list** structure using assembly language. Your program should read the data and operation flag from the input data set arrays and perform an operation in each System Tick ISR. The System Tick Timer will stop if the program reads all data in the input datasets. In order to organize the memory, you must write your own malloc() and free() functions. Detailed information about functions and constraints are given below.

Functions:

a) **__main()**: This function is already given to you. You must not change anything in the main function. The main function clears allocation table, clears error log table, initiates the global variables, and initializes the System Tick Timer using relevant functions. Then, it checks the program's status. The program goes to the STOP label if all data operations are completed. Table 1 shows the flag codes of the program status variable.

Table 1: Flag Codes of the Program Status.

Flag Code	Status
0	Program started.
1	Timer started.
2	All data operations finished.

b) **SysTick_Handler():** System Tick Handler function increases the tick count, reads the input data and its flag, and does operation of the input. Table 2 shows operation flags and explanation of the input data set operation flags.

Table 2: Operation Flags.

Flag Code	Operation
0	Remove the data from the linked list.
1	Insert the data to the linked list.
2	Transform the linked list to the array (The data will be ignored).

The function writes the error log to the memory if an error occurs in the operation. Table 3 shows the error codes and their explanations.

Table 3: Error Codes.

Error Code	Operation	Explanation
0	All Operations	No error.
1	Insertion	There is no allocable area. (The linked list is full.)
2	Insertion	Same data is in the array. (Duplicated insertion operation.)
3	Deletion	The linked list is empty.
4	Deletion	The element is not found.
5	Linked List to Array	The linked list could not be transformed. (The linked list is empty.)
6	-	Operation is not found.

At the end of the ISR, you should check whether all data is read or not. You should stop the timer if all input data is read.

Note: Assume that number of input data elements and input data flag elements is equal.

c) **SysTick_Init():** This function initializes System Tick Timer registers, start the timer, and update the program status register. The clock frequency of the CPU and the period of the System Tick Timer for your group are provided in the comments at the "**Forming Term Project Groups**" grade in the Ninova (in the grades section). Please check this information from the Ninova.

d) **SysTick_Stop()**: This function stops the System Tick Timer, clears the interrupt flag of it, and updates the program status.

e) **Clear_Alloc()**: Clear Allocation Table function clears all bits in the allocation table. In the beginning of the program, all bits of the allocation table must be cleared. Otherwise, some unused areas of the memory may appear to be used.

f) **Clear_ErrorLogs():** Clear Error Logs function clears all cells in the Error Log Array because some memory addresses contain garbage values.

g) **Init_GlobVars():** Init Global Variables function initializes global variables. All global values must be zero initially. Table 4 shows the global variables and their usage purpose.

Table 4: Global Variables.

Name	Usage
TICK_COUNT	Stores how many times the system tick timer generates interrupt.
FIRST_ELEMENT	Stores the first element address of the linked list.
INDEX_INPUT_DS	Stores the index of the input dataset to be read in the ISR.
INDEX_ERROR_LOG	Stores the index of the error log array for the new error log.
PROGRAM_STATUS	Stores the program status.

h) **Malloc()**: Malloc function finds the unused memory node and allocates it using the allocation table. The allocation table consists of 20 words. (It can be increased or decrease in the evaluation.) Each word has 32 bits and each bit points to one memory node. If the bit is 0, the memory node is not allocated. Otherwise, it is allocated as in Figure 1. The function returns the address of the allocated cell. It returns 0 value as an error if it cannot allocate any area, meaning that the linked list is full.

Each memory node consists of 1-word data and 1-word address area. The data area stores the value of the element. The address area points to the next element of the linked list.

Malloc function returns the memory address of the first possible memory node. Figure 1 shows an example of memory allocation. Here, the malloc function sets the 3rd bit of the first line of the allocation table and returns the address of the 3rd data.

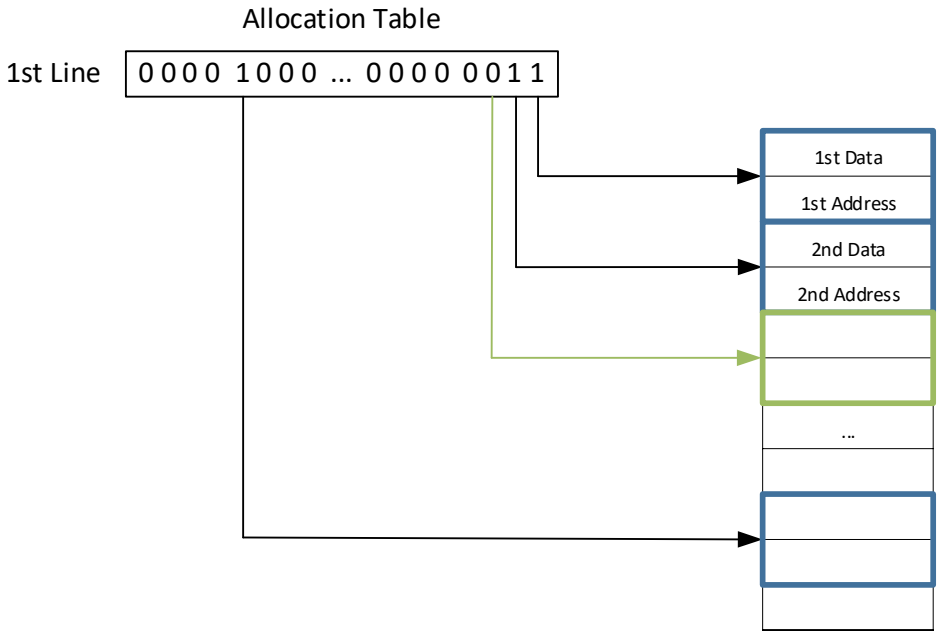

Figure 1: Example memory structure.

i) **Free(address)**: Address value is the address of the memory node. The free function clears the corresponding bits of the allocation table.

j) **Insert(value)**: The insert function gets a value as an argument. FIRST_ELEMENT variable stores the first element address of the linked list. Data is placed in the order on the list (not in memory). The first element of the list is the **smallest** data; the last element of the list is the **largest** data. The address pointer of the last element is zero, which means the null pointer. The linked list does not contain **duplicate values**. The function returns an error code if the linked list is full or the operation attempts to add a value that is already in the linked list. Otherwise, the function returns the success code.

Figure 2 shows an example of an insertion operation. #15 should be placed between the elements of #10 and #20. Then, the next pointer of the first element should point to the newly inserted element, thus the next pointer of the new element points now to the 3rd element of the linked list, which has #20.

Note: You must use the malloc function to create a new element. You must only change the pointers of the linked list to add new value. You do not need to change any value of the other elements.

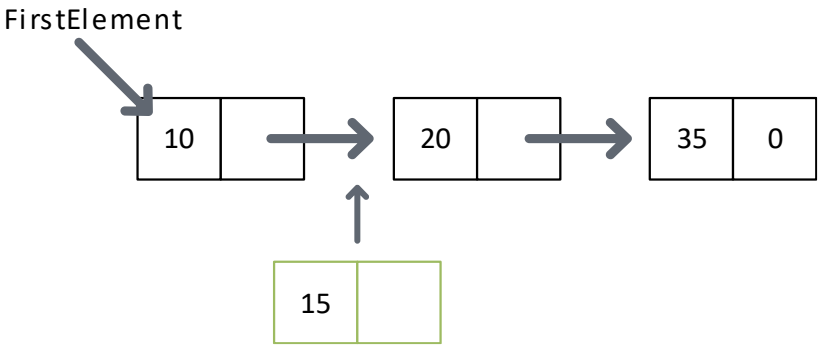

Figure 2: Example insertion operation.

k) **Remove(value)**: The remove function finds the elements having the same data given in the argument value. Then, it removes this element. The function returns an error code if the linked list is empty or the element is not in the linked list. Otherwise, the function returns the success code.
Figure 3 shows an example for remove operation. At the end of the function, the element having #20 is removed, the next pointer of the previous element points now to the next element which has #35, the function returns 0 value (There is no error).
Note: You must clear the corresponding bits of the element from the allocation table.

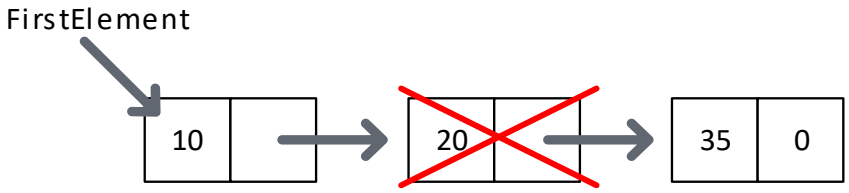

Figure 3: Example deletion operation.

l) **LinkedList2Arr()**: The LinkedList2Arr function clears the content of the DATA_MEM area, then transforms the linked list to the array. The values of the array will be stored in the DATA_MEM area. The function returns the error code if the linked list is empty. Otherwise, it returns the success code. Figure 4 shows an example of a sorted set linked list. At the end of the function, DATA_MEM area will contain [#10, #20, #35] values for this example.

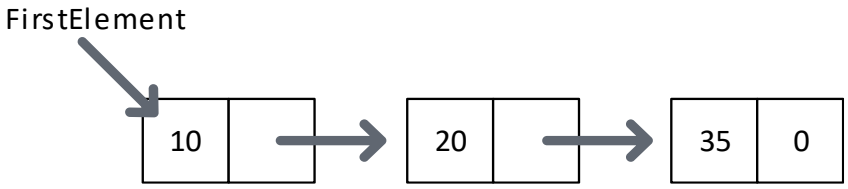

Figure 4: Example sorted set linked list.

m) **WriteErrorLog(Index, ErrorCode, Operation, Data):** Write Error Log function stores the error log of the input dataset operations. It takes Index, ErrorCode, Operation, and Data variables as the arguments. Then, it stores these data to the current available area of the LOG_MEM if the LOG_MEM array is not full. Table 5 shows Error Log Struct to store in the LOG_MEM area.

Table 5: Error Log Struct.

Size	Variable	Explanation
16-bit	Index	Index of the input dataset.
8-bit	ErrorCode	Error code of the operation.
8-bit	Operation	Operation of the current input data.
32-bit	Data	Current data to operate.
32-bit	Timestamp	Current System Tick Timer working time (in microseconds).

n) **GetNow():** The GetNow function returns the working time of the System Tick Timer in microseconds.
Hint: You can calculate actual working time using the TICK_COUNT variable and System Tick Timer Current Value Register. You do not need to use the Calibration Register.
Note: You assume that the program does all operations in less than 70 minutes.

Constraints:

- You must implement all functions in the program using the same function name, parameters, and return value type.
- You have to use the program template of the term project.
- You mustn't use any new area for the allocation table, linked list, array, and error log except the given area in the template.
- The numbers of the input data, allocation table, linked list, array, and error log can be changed in the test. You shouldn't use a constant number to detect the end of the area instead of using the endpoint label.
- You must use the R0-R3 as arguments and R0 as return value registers.
- Your main function name or label must be "__main".
- You should not change the main function.
- You must write your code between "USER CODE BEGIN" and "USER CODE END" phrases. You should not make any changes out of this area.
- Your code should include a comment for each line. Otherwise, points will be deducted.
- If you need to use any additional value or address, please specify it in comment lines.
- If you need to calculate any value, please write your formula and explain it step by step as a comment in the code and the report.
- The program must be implemented with Arm Cortex M0+ assembly language.
- Your assembly source file is expected to work with Keil µVision IDE v5.
- Default configuration must be sufficient to run your programs. If your program expects any different configuration parameter, please write this at the top of the code in comment lines.
- You must write a report for this project. The template of the reports is uploaded with homework.
- Your reports must be written in Latex format. You can use any Latex editor whichever you want. If you upload your report without a Latex file, you directly get 0 as your report grade. You should upload both .pdf and .tex files of your report.
- One submission for each group is enough. Please do not make the term project to be submitted more than once by another member of the group.

Submission: Please submit 1) an assembly file for the implementation of the program, 2) startup file, 3) report file, and 4) source code of the report. Type group number, your names, and student IDs at the top of the file as comments. You are expected to submit your term project through the Ninova system before the due date. Late submissions will not be accepted.

Any solution must be your own work. If any plagiarism is detected, disciplinary regulations of the university will be followed.

Grading: Code of the project: 25%, Report: 15% (over term average).

Note: If you have any questions regarding the exam, you may contact to teaching assistant of the course (kadir.ozlem@itu.edu.tr).