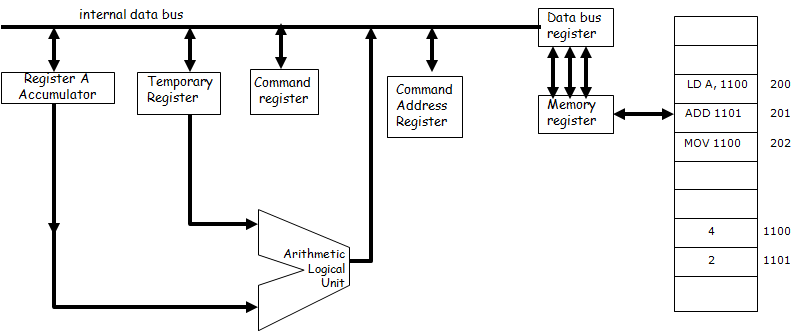
**RULES**

1. Assignments will NOT be graded if submitted after the due date. No exemptions will be given.
2. Copying code from any other solution (internet, other students, etc.) will automatically receive 0 for that assignment.
3. Giving code to anyone will automatically receive 0 for that assignment.
4. If a student receives 0 from two homework for any reason (not done, cheated, gave the code to another student, etc…), all six assignment grades will be given 0.
5. A report will be written for each assignment. Your report should include the following:
   1. The problem definition
   2. Explanation of the algorithm(s) used to solve that problem
   3. The flowchart of each module developed
   4. Complete source code ready for compilation
   5. Output instances and the input for that particular output
   6. Execution instructions (so I can execute your program)

**Project 1: Computer Architecture**

This is a very simple computer system that has main memory, accumulator, ALU an internal bus and an external bus as illustrated in the following figure.



ADD, SUBT, MULT, DIV, LD, and MOV are the only operations supported.

The input for writing to the memory should be (memory address, value): ex: 200, LD A, 1100 should place the instruction LD A, 1100 to memory location 200.

The last input should be (CAR, memory address) which should write the memory address to command address register.

After that the execution of the command should take place as discussed in the class

At the minimum, the steps of the execution and changes in any part (registers, memory, etc) should be output as text (80 points)

If you implement a graphical interface as shown above and illustrate the execution on this graphical interface, you will receive additional 20 points.

DUE DATE: February 26, 2014; 23:30 pm

E-mail your report to raif.onvural@izmir.edu.tr

**Project 2: Multiprogramming**

In this assignment, calculate the minimum time to execute N processes (2<=N<=4) in parallel. The input to the program is N, terminal time, disk time, printer time, total memory size. In addition, for each process define the sequence of operations in the process as input:

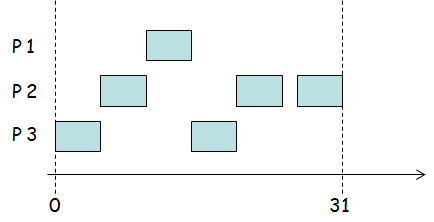
(CPU time, I/O type, CPU time, I/O type, CPU time, I/O type, CPU time, I/O type, ...) as a linked list.

Ex: (3, terminal, 4, terminal, 2, terminal) means run the CPU for 3 ms, request terminal operation, CPU for 4 ms, request terminal operation, CPU for 2 ms, request terminal operation).

Note that the sequence you run processes determines the total time to execute all N processes. Your program should find the sequence that results in minimum total time.

The output at the minimum should illustrate the sequence of events for all processes and the total time it takes to complete all N processes. (80 points).

If you implement a graphical interface as shown below and illustrate the execution on this graphical interface, you will receive additional 20 points.



DUE DATE: March 12, 2014; 23:30 pm

E-mail your report to raif.onvural@izmir.edu.tr

**Project 3: CPU Scheduler**

In this assignment, you will write a program to schedule N processes (2<=N<=5). Your program will support the following scheduler types.

FCFS, preemptive SJF, non-preemptive SJF, and round robin.

The input is the (arrival time, CPU time) for each process. For round robin scheduler, quantum is also input.

The output should display the sequence of executing the processes, the total time, waiting time for each process, and the average waiting time for all processes. (80 points)

If you implement a graphical interface and illustrate the execution on this graphical interface, you will receive additional 20 points.

DUE DATE: March 26, 2014; 23:30 pm

E-mail your report to raif.onvural@izmir.edu.tr

**Project 4: Memory Allocation and Garbage Collection**

In this assignment, write a program that will illustrate the process of allocating memory to processes. The input to the program is the total amount of memory available and the total number of processes that will be simulated. The memory requirements for each process is generated in random uniformly between 0.1\*MS and 0.4\*MS where MS is the memory size. In addition, the amount of time a process will stay in the memory is also generated in random.

N is the total number of processes that will be simulated.

MS is the memory size.

Generate the memory size requirement and the amount of time the process will stay in the memory for each process and place them in a FIFO as they are generated. Start running your simulation after all N processes are placed in the memory.

When the total amount of free memory is greater than or equal to the next process in the first place in the queue AND there is not a single contiguous memory available to place the process, perform garbage collection and continue.

The output should at least include the allocated and free memory space in the memory after each process is placed and a process leaves the memory. (80 points)

If you implement a graphical interface and illustrate the execution on this graphical interface, you will receive additional 20 points.

DUE DATE: April 9, 2014; 23:30 pm

E-mail your report to raif.onvural@izmir.edu.tr

**Project 5: Page Replacement**

In this assignment, you will write a program to do page replacement in a memory with N pages (N<=5) and M processes (M<=10). Your program will support the following algorithms.

Optimum, First In First Out, Least Recently Used, The Clock Page.

The input is the number of pages in the memory N, the number of processes M, and the sequence of page requests.

The output should display the processes in the main memory at the end of each page request. (80 points)

If you implement a graphical interface and illustrate the execution on this graphical interface, you will receive additional 20 points.

DUE DATE: April 22, 2014; 23:30 pm

E-mail your report to raif.onvural@izmir.edu.tr

**Page 6: Bakery Algorithm**

In this assignment, you will implement the Bakery’s algorithm for deadlock avoidance.

The input to the program is the number of processes, number of different resources, claim matrix, allocation matrix, and resource vector.

For a given request vector, the program will determine if these can be given to the requesting process or not. The output will display the state (safe or unsafe). If safe, it should output the sequence of running the processes to the safe state. If unsafe, the output should illustrate that all possible executions result in deadlock. (80 points)

If you implement a graphical interface and illustrate the execution on this graphical interface, you will receive additional 20 points.

DUE DATE: May 7, 2014; 23:30 pm

E-mail your report to raif.onvural@izmir.edu.tr