***King Fahd University of Petroleum and Minerals***

***College of Computer Science and Engineering***

***Computer Engineering Department***

**COE 306: INTRODUCTION TO EMBEDDED SYSTEMS**

**Term 161 (Fall 2016-2017)**

**Major Exam II**

**Saturday Dec. 10, 2016**

**Time: 150 minutes, Total Pages: 10**

**Name:\_KEY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_\_\_\_\_\_\_**

**Notes:**

* Do not open the exam book until instructed
* Answer all questions
* All steps must be shown
* Any assumptions made must be clearly stated

|  |  |  |
| --- | --- | --- |
| **Question** | **Max Points** | **Score** |
| **Q1** | **26** |  |
| **Q2** | **10** |  |
| **Q3** | **10** |  |
| **Q4** | **8** |  |
| **Q5** | **6** |  |
| **Total** | **60** |  |

Dr. Aiman El-Maleh

# **[26 Points]**

# **(Q1)** Fill in the blank in each of the following questions:

## Two CPU power saving strategies are: reduce power supply voltage, run at lower clock frequency, disable function units with control signals when not in use, disconnect parts from power supply when not in use.

## DMA allows transfers between the CPU and I/O devices without involving the CPU.

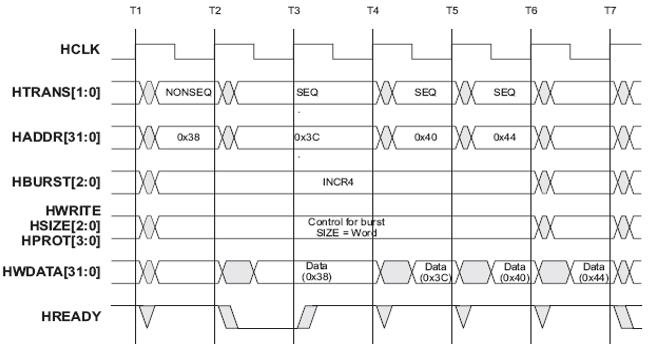
## In a microprocessor system, a high speed bus is connected to a low speed bus using a bridge.

## In AMBA, the UART is connected to the APB bus.

## In AMBA, an AHB transfer consists of an address phase, which lasts for one cycle(s) followed by a data phase, which lasts for one or more cycle(s).

## In AMBA basic AHB transfer, a slave can insert wait states by setting the HREADY signal to low.

## Complete the given timing diagram for an AMBA AHB pipelined burst write transfer:



## Complete the given timing diagram for an AMBA APB write transfer:

## 

## Given two ARM object files, file1 and file 2 of sizes of 0x100 and 0x60 bytes respectively. Suppose that file1 references a symbol X in file2, which is at offset 0x20 in file2. Suppose that the two object files are linked with file1 starting at address 0x400000 followed by file2. Then the resolved address of symbol X in file2 will be 0x400120.

## Two disadvantages of static linking are producing big executable files (all/most of needed libraries inside) and they don’t benefit from updates to library.

## The order in which a compiler-generated code will execute operations in the following statement is 1 3 2 6 5 4.

x = (a + b) \* (c - d) - e / (f-7)

1 2 3 4 5 6

## Given the following loop:

## for (i=0; i<9; i++)

## a[i] = b[i] + c[i];

## The partially unrolled loop of the given loop such that in each iteration three statements are executed is:

## for (i=0; i<9; i+=3){

## a[i] = b[i] + c[i];

## a[i+1] = b[i+1] + c[i+1];

## a[i+2] = b[i+2] + c[i+2];

## }

OR

## for (i=0; i<3; i++){

## a[i\*3] = b[i\*3] + c[i\*3];

## a[i\*3+1] = b[i\*3+1] + c[i\*3+1];

## a[i\*3+2] = b[i\*3+2] + c[i\*3+2];

## }

## Loop tiling breaks a loop into a set of nested loops for the objective of enhancing cache reuse.

## Considering the C array declaration int x[100][200], the byte offset of element x[10][5] is 10\*200\*4+5\*4=8020.

## Given a direct mapped cache of 512 lines with each line storing 8 bytes, to reduce cache conflicts while executing the given code, the size of pad array (i.e., x) should be 2.

We need to insert 8 bytes to make the index of array b different from array a. Thus, we need to insert two integers=8bytes.

## int a[1024];

## int pad[x];

## int b[1024];

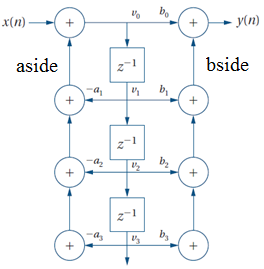
## int sum=0;

## for (i = 0; i < 1024; i++)

## sum += a[i] \* b[i];

**[10 Points]**

# **(Q2)** It is required to implement the following IIR filter using a circular buffer:



# Define the necessary variables for declaring the circular buffer.

# Show the C code for the function init( ) for initializing the buffer.

# Show the C code for the function put( ) for adding a new value to the buffer.

# Show the C code for the function get( ) for getting a value from the buffer. The function gets the ith value from the circular buffer with zero being the latest value put in the buffer.

# Show the C code for the function iir( ) that receives a new value x(n) and returns a computed value y(n) . Assume that the coefficients a and b are stored in two integer arrays as given below.

int a[4] = {0, a1, a2, a3};

int b[4] = {b0, b1, b2, b3};

HINT: Compute aside and bside variables first and then compute v0 and y. It is sufficient to store only 3 values in the buffer.

# #define SIZE 3

# int buffer[SIZE];

# int pos;

void init() {

for (int i = 0; i < SIZE; i++)

buffer[i] = 0;

pos = SIZE - 1;

}

# void put(int value) {

# pos = (pos + 1) % SIZE;

# buffer[pos] = value;

# }

# /\* get the ith value from the circular buffer; zero being the newest value \*/

# int get(int i) {

# int index = (pos - i) % SIZE;

# return buffer[index];

# }

# int iir(int xnew) {

# int i, t, aside=0, bside=0, v0, result;

# for (i=0; i<SIZE; i++){

t = get(i);

# aside += -a[i+1] \* t;

# bside += b[i+1] \* t;

# }

v0 = xnew + aside

# result = b[0] \* v0 + bside;

# put(v0); /\* put the new value in \*/

# return result;

# }

**[10 Points]**

# **(Q3)** Consider the data block given below:

x = a + b;

w = c + d;

y = x \* b;

y = y + a;

z = w \* y;

# Rewrite the given data block in single-assignment form, and then draw the data flow graph for that form.

y1

y

x

a

b

c

d

w

z

x = a + b;

w = c + d;

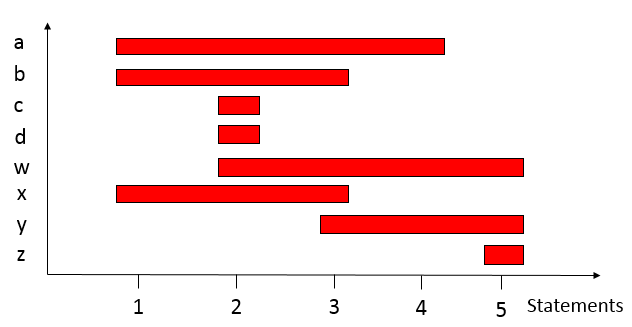
y = x \* b;

y1 = y + a;

z = w \* y1;

# Determine the minimum number of registers required to perform the operations when they are executed in the order shown in the code. Show the lifetime graph.

Life Time Graph:



Number of registers = 6.

# Determine the order of execution of operations that gives the smallest number of required registers and state the number of registers required. Show the lifetime graph.

We reorder the statements as follows to minimize the number of needed registers.

x = a + b;

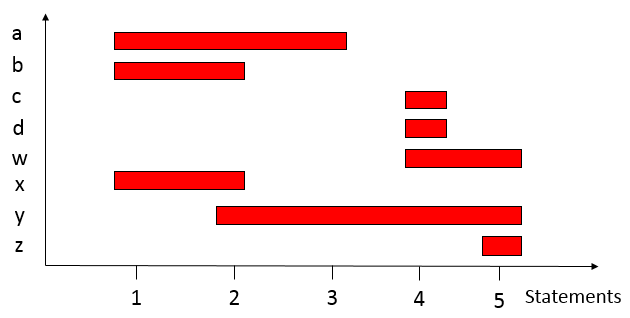
y = x \* b;

y = y + a;

w = c + d;

z = w \* y;

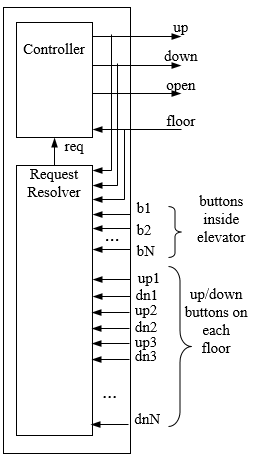
Life Time Graph:



Number of registers = 4.

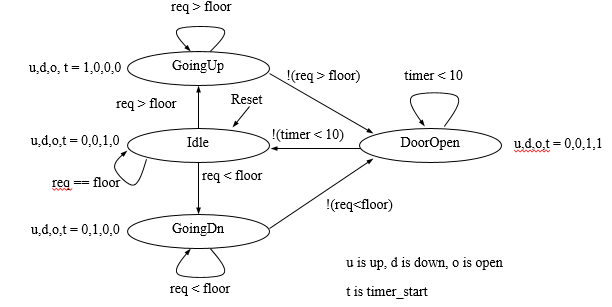
**[8 Points]**

**(Q4)** An elevator system of an N-Story building is composed of two units, a controller unit and a request resolver unit as shown below. The request resolver unit receives N buttons inside the elevator to indicate the desired floor. It also receives two buttons from each floor one for going up and another for going down requests. In addition, it receives a floor input indicating the current floor level and the two signals up and down indicating the direction of movement of the elevator. The controller unit receives the requested floor from the request resolver unit and a floor input indicating the current floor level. It generates three control signals as outputs: up, down and open. The signals up and down control the direction of movement of the elevator while the open signals control the elevator door.



The system will move the elevator either up or down to reach the requested floor. Once at the requested floor, the elevator door is opened for at least 10 seconds, and is kept open until the requested floor changes. It must be ensured that the elevator door is never open while the elevator is moving.

Draw the state machine diagram for the elevator controller unit. Design the controller using Moore model. Assume that the initial state is when the elevator is neither going up or down and the door is open. In your FSM, use the symbols U, D, O, T to represent elevator going up, elevator going down, door is open, and start timer, respectively



**[6 Points]**

**(Q5)** A sensor generates 10000 bytes of data every 10ms, and sends them to the CPUover a 2-MHz peripheral bus. The bus is 32-bit wide, and each bus transfer on the bus takes 3 bus cycles, 1 of which is used to transfer data, and the remaining cycles are used by the bus protocol.

# What is the CPU’s time budget (in seconds) for processing each 10000 bytes of sensory data?

Tbus = (D +O) N / W = (3) 10000 / 4 = 7500 bus cycles

tbus = T P = T / f = 7500 / (2 x 106) = 3750 μs = 3.75 ms

Then, the time budget available to the CPU for processing each 10000 bytes is:

tCPU = 10 – 3.75 = 6.25 ms

# If the CPU needs 30 cycles to process each byte of sensory data, what is the lowest frequency at which the CPU can run in order to process the data in time?

For each 10000 bytes:

TCPU = 30 x 10000 = 300,000 cycles

fCPU = TCPU / tCPU = 300,000 / 6.25 x 10-3 = 48,000,000 Hz = 48MHz