June 24, 2009

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Final Exam

Second Semester (082)

Time: 7:00 -9:30 PM

Student Name : \_KEY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID. : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Question** | **Max Points** | **Score** |
| **Q1** | **18** |  |
| **Q2** | **10** |  |
| **Q3** | **72** |  |
| **Total** | **100** |  |

Dr. Aiman El-Maleh

# **[18 Points]**

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

## Interrupts can be initiated by by both software and hardware while procedures can be initiated by software.

## Interrupts are identified by numbers.

## Interrupt types include software interrupts, hardware interrupts, processor interrupts or exceptions.

## Exceptions can be classidfied into faults, traps, aborts.

## There are 256 different interrupts.

## Interrupt flag controls whether maskable interrupts are delayed or not.

## In real mode, the address of the interrupt service routine of interrupt 21h is obatined from offset 4\*21h=84h within interrupt descriptor table (IDT).

## In protected mode, when an interrupt occurs, the following actions are taken by the CPU:

## Push EFLAGS register onto the stack,

## Clear interrupt and trap flags to disable further interrupts

## Push CS register onto the stack,

## Push EIP register onto the stack,

## Load CS register with 16-bit at memory address from IDT

## Load EIP register with 32-bit from IDT

## When the IRET instruction is executed in protected mode, the CPU performs the following steps:

## Pop 32-bit from the top of the stack into the EIP register,

## Pop 16-bit from the top of the stack into the CS register,

## Pop 32-bit from the top of the stack into the EFLAGS register.

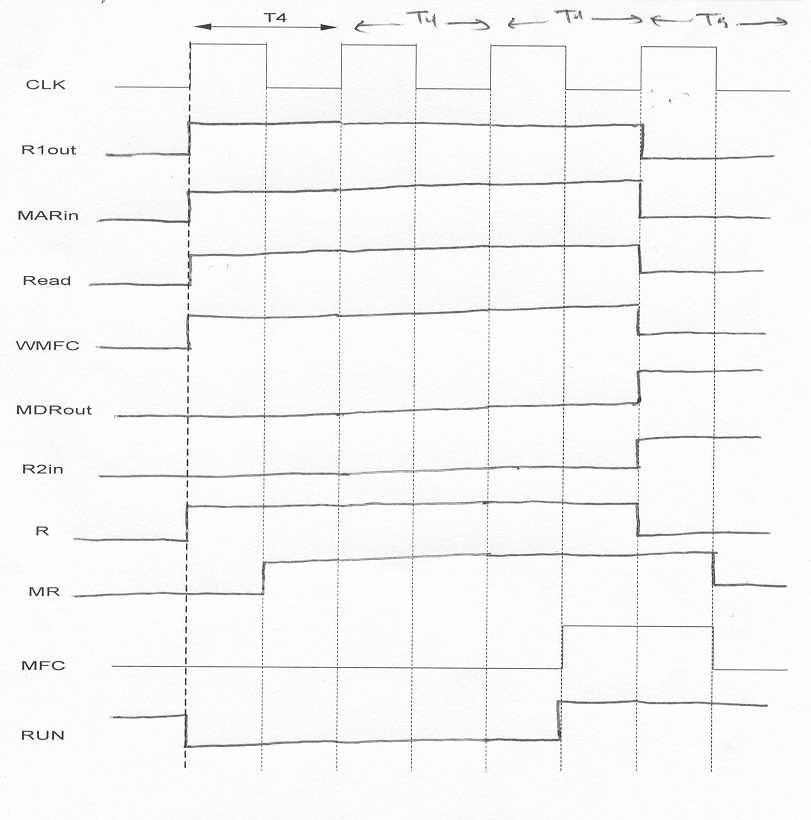
# **[10 Points]**

# **(Q2)** Given the CPU-Memory interface circuit shown below, complete the given timing diagram assuming the following given control sequnce as shown below:

**T4 R1out, MARin, READ, WMFC**

**T5 MDRout, R2in**

Note that the control unit is assumed to be rising-edge triggered.

# 

# **[72 Points]**

# **(Q3)** It is required to design an 8-bit CPU the has four 8-bit registers, namely R0, R1, R2, and R3. Assume that the data bus and address bus are 8-bits. The CPU has 8-bit instructions with the following foramt:

|  |  |  |
| --- | --- | --- |
| 4 bits | 2 bits | 2 bits |
| OPCODE | Rdst | Rsrc |

# The CPU has the following set of instructions:

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Register Transfer** | **Impact on ZF** |
| MOV Rdst, Rsrc | Rdst 🡨 Rsrc | Unchanged |
| XCHG Rdst, Rsrc | Rdst 🡨 Rsrc; Rsrc 🡨 Rdst | Unchanged |
| ADD Rdst, Rsrc | Rdst 🡨 Rdst + Rsrc | Changed |
| SUB Rdst, Rsrc | Rdst 🡨 Rdst – Rsrc | Changed |
| INC Rdst | Rdst 🡨 Rdst + 1 | Changed |
| DEC Rdst | Rdst 🡨 Rdst – 1 | Changed |
| SHL Rdst, Rsrc | Rdst 🡨 Rdst << Rsrc | Changed |
| SHR Rdst, Rsrc | Rdst 🡨 Rdst >> Rsrc | Changed |
| JMP Rdst | IP 🡨 Rdst | Unchanged |
| JZ Rdst | If (ZF=1) IP 🡨 Rdst | Unchanged |
| JNZ Rdst | If (ZF=0) IP 🡨 Rdst | Unchanged |
| LOOP Rdst, Rsrc | Rsrc 🡨 Rsrc-1; If (Rsrc≠ 0) IP 🡨 Rdst | Changed |

## Design a **three-bus data path** for this CPU. Clearly illustrate all design details and all the required control signals. Assume that the CPU has only a Zero Flag (ZF) affected by the executed instructions as indicated in the above table. Assume that you can only use the following combinational logic blocks in addition to basic gates like AND, OR, INV, XOR, MUX and Tri-state Buffers:

## An 8-bit **Adder** that has the inputs A[7:0], B[7:0] and Cin, and produces the Sum in C[7:0] and Cout.

## An 8-bit **Shifter** that has the inputs A[7:0] for specifiying the input to be shifted, B[2:0] to specify the amount of shift to be performed and ShiftDir to determine whether to shift left (ShiftDir=0) or shift right (ShiftDir=1). The shifted operand is produced on the output C[7:0].

## Design the data path to minimize the number of clock cycles needed for the execution of the specified instructions.Show the impact on the ZF in your design.

## Write the minimum number of control steps required for fetching an instruction from memory for this CPU, assuming asynchronous memory transfer.

## Write the minimum number of control steps required for the execution of each of the following instructions:

### XCHG Rdst, Rsrc

### SUB Rdst, Rsrc

### INC Rdst

### SHR Rdst, Rsrc

### JNZ Rdst

### LOOP Rdst, Rsrc

## Determine the **maximum speed** at which your CPU design will work given the following delay parameters:

|  |  |  |
| --- | --- | --- |
| **Name** | **Parameter** | **Delay** |
| Tri-state buffer delay | Tbuff | 50 ps |
| INV delay | Tinv | 40 ps |
| AND delay | Tand | 100 ps |
| OR delay | Tor | 100 ps |
| MUX delay | Tmux | 150 ps |
| XOR delay | Txor | 150 ps |
| Bus propagation time | tbp | 400 ps |
| ADDER delay | tALU | 500 ps |
| Shifter delay | tshifter | 300 ps |
| Flip-flop propagation time | tff | 150 ps |
| Flip-flop setup time | tsu | 100 ps |
| Flip-flop hold time | th | 50 ps |

## 

## Show the block diagram of the **hardwired control unit** organization for this CPU indicating all the necessary components and signals. Clearly indicate the size of the various components. Derive the equations for the END signal and all the signals that control register R1.

## Show the block diagram of the **microprogrammed control unit** organization for this CPU indicating all the necessary components and signals. Clearly indicate the size of the various components. Show the format and size of the control word. Show the microroutine of the JNZ instruction (show only the bits set to 1). Assume that the fetch microroutine is at address 0 and the address of JNZ microroutine is at address E. What is the **size of the control store** needed?

