June 16, 2010

COMPUTER ENGINEERING DEPARTMENT

COE 205

COMPUTER ORGANIZATION & ASSEMBLY PROGRAMMING

Final Exam

Second Semester (092)

Time: 7:00 -9:30 PM

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

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

|  |  |  |
| --- | --- | --- |
| **Question** | **Max Points** | **Score** |
| **Q1** | **15** |  |
| **Q2** | **8** |  |
| **Q3** | **22** |  |
| **Q4** | **40** |  |
| **Q5** | **15** |  |
| **Total** | **100** |  |

Dr. Aiman El-Maleh

# **[15 Points]**

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

## Interrupt is a mechanism by which a program's flow of control can be altered.

## The three main types of interrupts are: softwrae interrupts, hardware interrupts, processor interrupus.

## Software interrupts are mainly used in accessing I/O devices such as the keyboard, printer, screen, disk drive etc.

## Hardware interrupts can be either maskable or non-maskable.

##  The interrupt flag controls whether maskable interrupts are delayed or not.

## The CPU identifies Hardware Interrupt types by sending out an interrupt acknowledge (INTA) signal after which the interrupting device places the interrupt type number on the data bus.

## In order to have flexibility in storing ISRs in memory in any location, their addresses are stored in the interrupt descriptor table (IDT).

## In protected mode, when the CPU executes the instruction INT 21h, it performs following actions:

## 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 the 16-bit address from offset 10Ch from IDT

## Load EIP register with the 32-bit from offset 108h from IDT

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

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

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

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

## The CPU automatically generates a type 1 (single step) interrupt after executing each instruction if the trap flag is set.

# **[8 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 BXout, MARin, Read, WMFC**

**T5 MDRout, AXin**

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





# **[22 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** | **Effect on Flags** |
| IREG | R1🡨 0, R2🡨 0, R3🡨 0, R4🡨 0 | **N** |
| MOV Rdst, Rsrc | Rdst 🡨 Rsrc | **N** |
| XCHG Rdst, Rsrc | Rdst 🡨 Rsrc; Rsrc 🡨 Rdst | **N** |
| ADD Rdst, Rsrc | Rdst 🡨 Rdst + Rsrc | **Y** |
| SUB Rdst, Rsrc | Rdst 🡨 Rdst – Rsrc | **Y** |
| INC Rdst | Rdst 🡨 Rdst + 1 | **Y** |
| DEC Rdst | Rdst 🡨 Rdst – 1 | **Y** |
| SHL Rdst, Rsrc | Rdst 🡨 shift left Rdst by Rsrc | **Y** |
| SHR Rdst, Rsrc | Rdst 🡨 shift right Rdst by Rsrc | **Y** |
| Call Rsrc | Save IP address on Stack and JMP to Rsrc | **N** |
| Ret  | Restore IP from stack | **N** |
| JMP Rsrc | IP 🡨 Rsrc | **N** |
| JE Rsrc | If (ZF) IP 🡨 Rsrc | **N** |
| JNE Rsrc | If (!ZF) IP 🡨 Rsrc | **N** |
| JA Rsrc | If (!CF) IP 🡨 Rsrc | **N** |
| JB Rsrc | If (CF) IP 🡨 Rsrc | **N** |

## Design a **data path** for this CPU. Clearly illustrate all design details and all the required control signals. Design the data path to minimize the number of clock cycles needed for the execution of the specified instructions. 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 one select line SS to determine the required operation as follows: SS=0 to shift left, and SS=1 to shift right. The resulting operand is produced on the output C[7:0]. The last bit shifted is produced on the output signal LS.

## Assumat that R3 is the stack pointer. Assume that the CPU has a flags register consisting of 2 flags: carry flag and zero flag. Assume that flags are affected as indicated in the table above.



**[40 Points]**

# **(Q4)** Assume that a CPU has **16 instructions** with an opcode of 4 bits, seven **16-bit** general purpose registers namely AX, BX, CX, DX, SI, DI and SP, a **16-bit data bus**, and a **16-bit address bus**. Assume that all the instructions are 16-bit. The CPU has an **Arithmetic and Logic Unit (ALU)** with inputs A and B and output C, that can perform any of the following functions shown below based on the three selection lines AS2, AS1, and AS0:

|  |  |
| --- | --- |
| AS2 AS1 AS0 | Operation |
| 000 | C=A+B |
| 001 | C=A-B |
| 010 | C=A+1 |
| 011 | C=A-1 |
| 100 | C=B |
| 101 | C=A+2 |
| 110 | C=A-2 |
| 111 | C=NOT A |

# The CPU has also a **Shift Unit** that can perform shifting as shown below based on the two selection lines SS1 and SS0:

|  |  |
| --- | --- |
| SS1 SS0 | Operation |
| 00 | No shift |
| 01 | W=Shift logic right(X) by M bits |
| 10 | W=Shift logic left(X) by M bits |
| 11 | W=Shift arithmetic right(X) by M bits |

# Assume that the IR, PC, MAR, and MDR registers are also **16-bit regiters**. Also assume that the instructions for this processor follow the syntax and semantics and addressing modes of the 8086 processor. Also assume that registers play the same role as assumed in the 8086 processor.

# Assume that the data path of this processor is implemented using a **three-bus architecture** as shown in the next page.

## Write the minimum number of control steps required for fetching an instruction from memory.

T1 PCout, MARin,A, AS2, AS0 ALU(C=A+2), ALUout, PCin, Read, WMFC

T2 MDRout,B, AS2 ALU(C=B), ALUout, IRin

AX

BX

CX

DX

IR

PC

MAR

MDR

ALU

A

B

C

A Bus

B Bus

C Bus

**Data Path Design**

SI

DI

SP

TMP

SHIFTER

M

X

W

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

### MOV I, 10

T3 IR(offset),out,A, MARin,A, IR(const.),out,B, AS2 ALU(C=B), ALUout, MDRin, Write, WMFC

T4 End

### SUB AX, Array[BX]

T3 IR(offset),out,A, BX,out,B, ALU(C=A+B), ALUout, MARin,C, Read, WMFC

T4 AXout,A, MDRout,B, AS0 ALU(C=A-B), ALUout, AXin, End

### PUSH AX

T3 SP,out,A, AS2, AS1 ALU(C=A-2), ALUout, Spin, MARin,C

T4 AXout,B, AS2 ALU(C=B), ALUout, MDRin, Write, WMFC

T5 End

### RET N

T3 SP,out,A, MARin,A, Read, AS2, AS0 ALU(C=A+2), ALUout, Spin

T4 SPout,A, IR(const.),out,B, ALU(C=A+B), ALUout, SPin, WMFC

T5 MDRout,B, AS2 ALU(C=B), ALUout, PCin, End

### LOOPE Next

T3 CXout,A, AS1, AS0 ALU(C=A-1), ALUout, CXin, if (ZF=0) End

T4 PCout, IRout,B, ALU(C=A+B), ALUout, if (CX≠ 0) PCin, End

### DEC [BX]

T3 BXout,A, MARin,A, Read, WMFC

T4 MDRout,A, AS1, AS0 ALU(C=A-1), ALUout, MDRin, Write, WMFC

T5 End

### JC Next

T3 PCout, IRout,B, ALU(C=A+B), ALUout, if (CF=1) PCin, End

### SAR AX, 5

T3 AXout,A, IR(const.)out,B, SS1, SS0 Shifter(shift aritmetic right), Wout, AXin, End

## Based on the fetch and execution control sequence of the eight instructions given in this question, show the logic equation required for generating the signals **AXin** and **PCin.**

AXin = T4 SUB + T3 SAR

PCin = T1 + T5 RET + T4 LOOPE CXZ’ + T3 JC CF

Note that it is assumed here that the signal CXZ is 1 if the content of register CX is 0.

**[15 Points]**

#  **(Q5)** Using the CPU described in the previous question and the given data path, it is required to implement the control unit for this CPU using both hardwired and microprogrammed approaches. Assume that the number of signals that have to be generated by the control unit to control the data path and the memory interface circuit for this CPU is **40** signals.

## Assume that the maximum number of control steps required for the fetch and execution of any of the 16 instructions is **10 control steps**. Show the block diagram of the **hardwired control unit** organization for this CPU indicating all the necessary components and signals. Assume that the flags needed for the execution of the instructions are the zero and carry flags. Clearly indicate the size of the various components.



## Assume that the control unit for this CPU is to be implemented using a microprogrammed approach and that the number of different control words to be stored in the control store for all the microroutines is **100 CWs**. Also, assume that the microprogrammed control unit design should support unconditional branch (UBR), branch on zero (BRZ), branch on not zero (BRNZ), branch on carry (BRC) and branch on no carry (BRNC).

## Show the format and size of the control word using **horizontal control store**. What is the **size of the control store** needed?



## 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.

##