NSU Term Final Examination (Fall–2020)

CSE: 332 (Section-4)

Total Marks = 50; Time: 1 hour 45 Minutes

Answer all questions (5x10 = 50)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | A superscalar processor, shown below, uses “in-order-issue-in-order-completion” policy.  Consider a program having following sequence of instructions, where the syntax consists of an opcode followed by the destination register followed by one or two source registers. Instructions requiring floating Point Unit (FPU) are indicated in the comment field. Please note that LOAD, STORE and FPU operations require 3 clock cycles each. Integer operations require one clock cycle. Show the execution sequence of the program. Also calculate the run time if the CPU uses 1GHz clock.   |  |  | | --- | --- | | Instructions | Comment | | ADD R3, R1, R2 | FPU | | LOAD R6, [R3] |  | | AND R7, R6, R1 |  | | ADD R1, R6, R0 | FPU | | SUB R2, R1, R6 | FPU | | AND R3, R7, 15 |  | | LOAD R6, [R5] |  | | SUB R5, R3, R4 |  | | ADD R0, R1, R10 |  | | SHL R7, R0, 8 |  |   System  Fetch  Unit-1  Fetch  Unit-2  Decode  Unit-1  Decode  Unit-2  Integer ALU  Floating Point  Unit  Write-back  Unit-1  Write-back  Unit-2  Integer ALU |
| 2. | Assume that Machine-1, Machine-2 and Machine-3, use one-address, two-address and three-address instructions respectively. Consider a high-level language statement  P = (A × C +D) **/** (E × A - B × D) and comment on relative efficiency.  Use simple arithmetic, logical, store, load and conditional instructions. |
| 3. | Use STACK for storing the return address for a procedure return of the following program given below. Show the contents of stack following each CALL and RETURN instructions:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | Main Program   |  |  | | --- | --- | | Memory Address | Instructions | | 3000H |  | | ----- |  | | 3050H | CALL Proc 1 | | ---- |  | | 3080H | CALL Proc 2 | |  |  | |  |  | |  | END |   Procedure 1   |  |  | | --- | --- | | Memory Address | Instructions | | 5000H |  | | --- |  | | 5060H | CALL Proc 2 | |  |  | |  | RETURN |   Procedure 2   |  |  | | --- | --- | | Memory Address | Instructions | | 6000H |  | | --- |  | | 6030H | CALL Proc 3 | | ----- |  | | 6100H |  | |  |  | |  | RETURN | |   Procedure 3   |  |  | | --- | --- | | Memory Address | Instructions | | 1000H |  | |  |  | |  | RETURN |   …..   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | |  | |  | | 3051 | | -- | | |  | | --- | |  | | 5061 | | 3051 | |  | | |  | | --- | | 6031 | | 5061 | | 3051 | |  | | |  | | --- | |  | | 5061 | | 3051 | |  | | |  | | --- | |  | |  | | 3051 | |  | | | |  | | --- | |  | |  | |  | | -- | | |  | | --- | |  | |  | | 3081 | | -- | | |  | | --- | |  | | 6031 | | 3081 | |  | | |  | | --- | |  | |  | | 3081 | |  | | |  | | --- | |  | |  | | -- | | -- | | |
| 4. | The block diagram of a 6-bit multiplier with optimal size ALU (adder) and registers is given below.    Assume that the Multiplicand and Multiplier registers are loaded with 6-bit numbers for multiplication and the product register is cleared initially as shown below. Two 6-bit unsigned binary numbers are to be multiplied using shift or add/shift operations. During this process, show the contents of all these registers that change. Determine the contents of these registers after an add/shift or just shift operation and fill up the following table after each operation.   |  |  |  |  | | --- | --- | --- | --- | | Operation  Shift and/or Add | Product | Multiplicand | Multiplier | | Initial Values | 0 0 0 0 0 0 0 0 0 0 0 0 | 0 1 1 0 1 1 | 0 1 0 1 0 1 | |  |  |  |  | |  |  |  |  | |
| 5. | Show the 2’s complement multiplication using Booth’s algorithm  Multiplicand: +5 Multiplier: -7    Multiplicand: +5 Multiplier: -7  +5: 0101  -7: 1001(in 2’s complement)   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | CYCLE | OPERATIONS | Content of A | Content of Q | Q-1 | Comments, if any | Content of M | | Initial Value | Initialization | 0000 | 1001 | 0 | Q0Q-1= 10 | 0101 | | Cycle-1 | A= A – M | 1011 | 1001 | 0 | Q0Q-1= 10 | | Arithmetic Shift right  (A Q Q-1) | 1101 | 1100 | 1 | Q0Q-1= 01 | | Cycle-2 | A= A+M | 0010 | 1100 | 1 | Q0Q-1= 01 | | Shift right  (A Q Q-1) | 0001 | 0110 | 0 | Q0Q-1= 00 | | Cycle-3 | Shift right  (A Q Q-1) | 0000 | 1011 | 0 | Q0Q-1= 10 | | Cycle-4 | A=A-M | 1011 | 1011 | 0 | Q0Q-1= 10 | | Shift right  (A Q Q-1) | 1101 | 1101 | 1 | Q0Q-1= 11 | |  |  |  |  |  |  | | Product is in: AQ pair: 11011101 in 2’s complement since MSB is 1, to get magnitude, take 2’s complement of AQ: 00100010 + 1 = 00100011 = 35  Answer: - 35 | | | | | | |
|  | **Bonus question-1 (Marks-6, to be added with MT**)  Given the signals Ci (i = 1, 2, 3….13) in following diagram, list the activation of control signals in sequential order to execute  DIV M1 (Divide content of AC by the contents of memory location M1 and store quotient in M2 and remainder in M3 locations of memory)  Assume C16 is a control signal used to select division operation in ALU. Also use CR and CW signals to read from memory and write onto memory. Figure below to answer this question. |
|  | Bonus question-2 (Marks- 6, to be added with total)   |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Write the sequence of micro-operations and control signals required to process following instruction. Show Instruction fetch and execution using following table.  ADD M1; M1 is a memory address.  You can add more control signals for Memory and ALU operations, if required.   |  |  |  | | --- | --- | --- | | Time steps | Micro operations | Control signals | | T1 |  |  | | T2 |  |  | |  |  |  | |  | |

Tentative solution

|  |  |  |
| --- | --- | --- |
| Time steps | Micro operations | Control signals |
| T1 | MAR ←(PC) | C3, C4 |
| T2 | MBR ←Memory  PC ←(PC) + I | C5, CR |
| T3 | IR ←(MBR) | C6, C2 |
| T4 | MAR ← (IR(Address M1)) | CX(assume to transfer to bus)  C4 |
| T5 | MBR ← Memory(M1) | C5, CR |
| T6 | Y ← MBR | C6, C9 |
| T7 | Z = AC + Y | CADD |
| T8 | AC ← Y | C11, C7 |

NSU Term Final Examination (Fall–2020)

CSE: 332 (Section-5)

Total Marks = 50; Time: 1 hour 45 Minutes

Answer all questions (5x10 = 50)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | A superscalar processor, shown below, uses “in-order-issue-in-order-completion” policy.  Consider a program having following sequence of instructions, where the syntax consists of an opcode followed by the destination register followed by one or two source registers. Instructions requiring floating Point Unit (FPU) are indicated in the comment field. Please note that LOAD, STORE and FPU operations require 4, 4 and 3 clock cycles respectively whereas integer operations require one clock cycle only. Show the execution sequence of the program. Also calculate the run time if the CPU uses 2GHz clock.   |  |  | | --- | --- | | Instructions | Comment | | ADD R3, R1, R2 | FPU | | LOAD R6, [R3] |  | | AND R1, R6, R1 |  | | ADD R1, R6, R0 | FPU | | SUB R2, R1, R6 | FPU | | AND R4, R3, 15 |  | | LOAD R6, [R4] |  | | SUB R5, R3, R4 |  | | ADD R0, R1, R6 |  | | SHL R7, R0, 8 |  |   System  Fetch  Unit-1  Fetch  Unit-2  Decode  Unit-1  Decode  Unit-2  Integer ALU  Floating Point  Unit  Write-back  Unit-1  Write-back  Unit-2  Integer ALU   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | CLK | FU1 | FU2 | DU1 | DU2 | FPU | IALU1 | IALU2 | WB1 | WB2 | | 0 | I1 | I2 |  |  |  |  |  |  |  | | 1 | I3 | I4 | I1 | I2 |  |  |  |  |  | | 2 | I5 | I6 | I3 | I4 | I1 |  |  |  |  | | 3 | I7 | I8 | I5 | I6 | I1 |  |  |  |  | | 4 | I9 | I10 | I7 | I8 | I1 |  |  |  |  | | 5 |  |  | 19 | I10 |  |  |  | I1 |  | | 6 |  |  |  |  |  | I2 |  |  |  | | 7 |  |  |  |  |  | I2 |  |  |  | | 8 |  |  |  |  |  | I2 |  |  |  | | 9 |  |  |  |  |  | I2 |  |  |  | | 10 |  |  |  |  |  |  |  | I2 |  | | 11 |  |  |  |  |  | I3 |  |  |  | | 12 |  |  |  |  |  |  |  | I3 |  | | 13 |  |  |  |  | I4 |  |  |  |  | | 14 |  |  |  |  | I4 |  |  |  |  | | 15 |  |  |  |  | I4 |  |  |  |  | | 16 |  |  |  |  |  |  |  | I4 |  | | 17 |  |  |  |  | I5 | I6 |  |  |  | | 18 |  |  |  |  | I5 |  |  |  |  | | 19 |  |  |  |  | I5 |  |  |  |  | | 20 |  |  |  |  |  |  |  | I5 | I6 | | 21 |  |  |  |  |  | I7 | I8 |  |  | | 22 |  |  |  |  |  | I7 |  |  |  | | 23 |  |  |  |  |  | I7 |  |  |  | | 24 |  |  |  |  |  | I7 |  |  |  | | 25 |  |  |  |  |  |  |  | I7 | I8 | | 26 |  |  |  |  |  | I9 |  |  |  | |  |  |  |  |  |  |  |  | I9 |  | |  |  |  |  |  |  | I10 |  |  |  | |  |  |  |  |  |  |  |  | I10 |  | |
| 2. | Assume that Machine-1, Machine-2 and Machine-3, use one-address, two-address and three-address instructions respectively. Consider a high-level language statement  P = 1+ [(A +D× C) **/** (E × A - B × D)] and comment on relative efficiency.  Use simple arithmetic, logical, store, load and conditional instructions. |
| 3. | Use STACK for storing the return address for a procedure return of the following program given below. Show the contents of stack following each CALL and RETURN instructions:   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | Main Program   |  |  | | --- | --- | | Memory Address | Instructions | | 7000H |  | | ----- |  | | 7050H | CALL Proc 3 | | ---- |  | | 7080H | CALL Proc 2 | |  |  | |  |  | |  | END |   Procedure 3   |  |  | | --- | --- | | Memory Address | Instructions | | 3000H |  | | --- |  | | 3060H | CALL Proc 2 | |  |  | |  | RETURN |   Procedure 2   |  |  | | --- | --- | | Memory Address | Instructions | | 5000H |  | | --- |  | | 5030H | CALL Proc 1 | | ----- |  | | 5100H |  | |  |  | |  | RETURN | |   Procedure 1   |  |  | | --- | --- | | Memory Address | Instructions | | 8000H |  | |  |  | |  | RETURN |   …..   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  | | --- | |  | |  | | 7051 | | -- | | |  | | --- | |  | | 3061 | | 7051 | |  | | |  | | --- | | 5031 | | 3061 | | 3051 | |  | | |  | | --- | |  | | 3061 | | 7051 | |  | | |  | | --- | |  | |  | | 7051 | |  | | | |  | | --- | |  | |  | |  | | -- | | |  | | --- | |  | |  | | 7081 | | -- | | |  | | --- | |  | | 5031 | | 7081 | |  | | |  | | --- | |  | |  | | 7081 | |  | | |  | | --- | |  | |  | | -- | | -- | | |
| 4. | The block diagram of a 6-bit multiplier with optimal size ALU (adder) and registers is given below.    Assume that the Multiplicand and Multiplier registers are loaded with 6-bit numbers for multiplication and the product register is cleared initially as shown below. Two 6-bit unsigned binary numbers are to be multiplied using shift or add/shift operations. During this process, show the contents of all these registers that change. Determine the contents of these registers after an add/shift or just shift operation and fill up the following table after each operation.  **1000110111**   |  |  |  |  | | --- | --- | --- | --- | | Operation  Shift and/or Add | Product | Multiplicand | Multiplier | | Initial Values | 0 0 0 0 0 0 0 0 0 0 0 0 | 0 1 1 0 1 1 | 0 1 0 1 0 1 | | 1: SR then ADD | 0 0 0 0 0 0 0 1 1 0 1 1 | 0 1 1 0 1 1 | 0 0 1 0 1 0 | | 2: SR | 0 0 0 0 0 0 0 1 1 0 1 1 | 0 1 1 0 1 1 | 0 0 0 1 0 1 | | 3: SR then ADD | 0 0 0 0 1 0 0 0 0 1 1 1 | 0 1 1 0 1 1 | 0 0 0 0 1 0 | | 4: SR | 0 0 0 0 1 0 0 0 0 1 1 1 | 0 1 1 0 1 1 | 0 0 0 0 0 1 | | 5: SR then ADD | 0 1 0 0 0 0 1 1 0 1 1 1 | 0 1 1 0 1 1 | 0 0 0 0 0 0 | |
| 5. | Show the 2’s complement multiplication using Booth’s algorithm  Multiplicand: +7 Multiplier: - 6     |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | CYCLE | OPERATIONS | Content of A | Content of Q | Q-1 | Comments, if any | Content of M | | Initial Value | Initialization | 0000 | 1010 | 0 | Q0Q-1= 00 | 0111 | | Cycle-1 |  |  |  |  |  | | Arithmetic Shift right  (A Q Q-1) | 0000 | 0101 | 0 | Q0Q-1= 10 | | Cycle-2 | A= A-M | 1001 | 0101 | 0 | Q0Q-1= 10 | | Shift right  (A Q Q-1) | 1100 | 1010 | 1 | Q0Q-1= 01 | | Cycle-3 | A=A+M | 0011 | 1010 | 1 | Q0Q-1= 01 | | Shift right  (A Q Q-1) | 0001 | 1101 | 0 | Q0Q-1= 10 | | Cycle-4 | A=A-M | 1010 | 1101 | 0 | Q0Q-1= 10 | | Shift right  (A Q Q-1) | 1101 | 0110 | 1 | Q0Q-1= 01 | |  |  |  |  |  |  | | Product is in: AQ pair: 11010110 in 2’s complement since MSB is 1, to get magnitude, take 2’s complement of AQ: 00101001 + 1 = 00101010 = 42  Answer: - 42 | | | | | | |
|  | **Bonus question-1 (Marks-6, to be added with MT**)  Given the signals Ci (i = 1, 2, 3….13) in following diagram, list the activation of control signals in sequential order to execute  LOAD M2  DIV M1 ;(Divide content of AC by the content of memory location M1 and store quotient in M2 and remainder in M3 locations of memory respectively)  Assume C16 is a control signal used to select division operation in ALU. Also use CR and CW signals to read from memory and write onto memory.    Tentative solution:   |  |  |  |  | | --- | --- | --- | --- | | time | Microoperations | Control signals | comments | | T1 | MAR ←(PC) | C2 | FETCH  (LOAD M2) | | T2 | MBR ←Memory  PC ←(PC) + I | C5, CR | | T3 | IR ←(MBR) | C4 | | T4 | MAR ← (IR(Address M2)) | C8 |  | | T5 | MBR ← M2 | C5, CR |  | | T6 | AC ← MBR | C10 |  | |  | MAR ←(PC) | C2 | FETCH  DIV M1 | |  | MBR ←Memory  PC ←(PC) + I | C5, CR |  | |  | IR ←(MBR) | C4 |  | |  | MAR ← (IR(Address M1)) | C8 |  | |  | MBR ← M1 | C5, CR |  | |  | ALU ←MBR | C6 |  | |  | DIV AC/DATA FROM M1 | C16 | ; QUOTIENT IN AC & REMAINDER IN TEMP REGISTER IN ALU | |  | MBR ← AC | C11 |  | |  | MAR ← M1 |  |  | |  | MEMORY ← MBR |  |  | |  | ----- |  |  | |
|  | Bonus question-2 (Marks- 6, to be added to total)   |  |  | | --- | --- | | Write the sequence of micro-operations and control signals required to **Fetch** and **Execute** following instruction. Follow the table, indicated below.  **MUL M1**; Multiply content of AC by data stored at memory address M1 and save in AC.  You can add more control signals for Memory and ALU operations, if required. |  | |

Tentative solution

|  |  |  |
| --- | --- | --- |
| Time steps | Micro operations | Control signals |
| T1 | MAR ←(PC) | C3, C4 |
| T2 | MBR ←Memory  PC ←(PC) + I | C5, CR |
| T3 | IR ←(MBR) | C6, C2 |
| T4 | MAR ← (IR(Address M1)) | CX(assume to transfer to bus)  C4 |
| T5 | MBR ← Memory(M1) | C5, CR |
| T6 | Y ← MBR | C6, C9 |
| T7 | Z = AC x Y | CM |
| T8 | AC ← Y | C11, C7 |