**Assignment - SMACO machine simulator Ready Reference**

A simple instruction Computer (SMACO) is a hypothetical machine with a small but effective instruction

set, can illustrate the design of simple software processors and program execution. The machine will incorporate the most commonly encountered hardware features and concepts and avoids the complexities. A simulator program is required to simulate the function of simple instruction computer such as fetching an instruction, decoding and executing it.

The hypothetical Simple machine (SMACO) has following features.

**Memory –** Memory consist of 6 digit words (decimal). Total size of memory is 1000 words (103) indicating the address size is 3 digits (address ranges from 0 to 999).

**Registers –** There are in all five registers four general purposes registers AREG, BREG, CREG and DREG numbered 1, 2, 3 and 4 respectively and one special purpose register LC – Location Counter which indicates the address of the instruction to be executed.

**Condition Codes** - There are SIX condition codes LT, LE, EQ, GT, GE and ANY numbered 1, 2, 3, 4, 5 and 6 respectively.

**Data Format** – Supports only six digit integer data stored in decimal form.

**Instruction Format** – It has single instruction format. Each instruction is of six digit length. The opcode, register operand and memory operand occupy 2, 1 and 3 digits in that order

opcode Register

operand

Memory

operand

Condition Codes

|  |  |  |
| --- | --- | --- |
| Condition code | Mnemonic | Description |
| 1 | LT -1 | Less than |
| 2 | LE -2 | Less than or equal to |
| 3 | EQ 0 | Equal to |
| 4 | GT 2 | Greater than |
| 5 | GE 1 | Greater than or equal to |
| 6 | ANY | Unconditional |

**Register Codes**

|  |  |  |
| --- | --- | --- |
| Condition code | Mnemonic | Description |
| 1 | AREG | A register |
| 2 | BREG | B register |
| 3 | CREG | C register |
| 4 | DREG | D register |

**Instruction Set**

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode | Mnemonic | Instruction | Operands |
| 00 | STOP | Stop or Halt execution | Operands unused |
| 01 | ADD | Add memory operand to register operand | Register and memory operand |
| 02 | SUB | Subtract memory operand from register  operand | Register and memory operand |
| 03 | MULT | Multiply memory operand to register  operand | Register and memory operand |
| 08 | DIV | Divide register operand by memory  operand | Register and memory operand |
| 04 | MOVER | Move memory operand contents to  register operand | Register and memory operand |
| 05 | MOVEM | Move register operand contents to  memory | Register and memory operand |
| 06 | COMP | Compare register and memory operands  to set condition code appropriately | Register and memory operand |
| 07 | BC | Branch to second operand depending on  condition code specified as first operand | Register and memory operand |
| 09 | READ | Read into memory operand | Only memory operand |
| 10 | PRINT | Print contents of memory Operand | Only memory operand |

It should be possible to load program from file into memory at specified location. File contains program as sequence of lines, each line containing address followed by content indicating the instruction to be stored at that address. The file ends with –1 followed by starting address indicating physical end of file.

|  |  |  |
| --- | --- | --- |
| Simple program to add two numbers | | |
| address | Content | Description |
| 100 | 090107 | Read into 107th memory address |
| 101 | 090108 | Read into 108th memory address |
| 102 | 041107 | Move contents of 107th memory address to register 1 |
| 103 | 011108 | Add contents of 108th memory address to register 1 |
| 104 | 051109 | Move contents of register 1 to memory address 109 |
| 105 | 100109 | Output contents of 109th memory address |
| 106 | 000000 | Halt – logical end of the program |
| 107 | 0 | Address to be used for first integer |
| 108 | 0 | Address to be used for second integer |
| 109 | 0 | Address to be used for result |

The above program should be stored in a file sum.sm as follows

|  |  |
| --- | --- |
| 100 | 90107 |
| 101 | 90108 |
| 102 | 41107 |
| 103 | 11108 |
| 104 | 51109 |
| 105 | 100109 |
| 106 | 0 |
| -1 | 100 |

Similarly, the program for printing factorial of the number read is given below. Store it in a file fact.sm.

|  |  |
| --- | --- |
| 100 | 090113 |
| 101 | 041113 |
| 102 | 042112 |
| 103 | 061112 |
| 104 | 071109 |
| 105 | 032113 |
| 106 | 021112 |
| 107 | 051113 |
| 108 | 075103 |
| 109 | 052114 |
| 110 | 100114 |
| 111 | 0 |
| 112 | 1 |
| -1 | 100 |

**Data Structure Design** – The SMAC0 machine has memory and a set of registers. Appropriate data structures need to be chosen to represent each one of them. Simulator program also need to store the last valid address in the physical file.

|  |  |
| --- | --- |
| **Component** | **Description** |
| Memory | An array of 1000 words each can store an integer |
| Program Counter | An integer indicating the address of instruction |
| General Purpose registers | Four general purpose registers numbered 1, 2, 3 and 4 |
| Condition Code register | A single integer with each bit representing a condition code or  An array of six registers each storing condition code separately |
| Last address | An integer indicating last valid address in physical file |

**Procedural Design** – The following table explains the input, algorithm and provides some implementation hints

|  |  |
| --- | --- |
| Procedure | Description |
| Simulator(main) | In a loop, Print options, Read an option, Branch depending on option |
| Load | Accept the filename, Open the file for reading, Read from the file |
| Print | Print from contents from pc to lc |
| Run | Input – start address, |
| Fetch the instruction, Decode the instruction While opcode is not zero  Depending on opcode take action  If opcode is 1 add memory operand to register  …  If opcode is 4 move contents of memory operand to register operand  …..  If opcode is 6 compare memory operand with register operand and set the condition code  If opcode is 7 take the jump to memory operand if condition code matches  If opcode is 9 read into memory operand …… |

Specifications Assignment No. 5

Write programs sum.sm and fact.sm in the following format of machine language instruction

<address> <instruction> e.g. 100 090200

Write a menu driven program to implement menus given in the assignment.

Handle the following scenarios

1. Menu options **b** and **c** should not get execute till the menu option **a** is executed.
2. Starting address or any address in the address or operand field should not be less than 0 or greater than 999.
3. Mnemonic opcode 1, 2, 3, 4, 5, 6 and 8 should not have register code either 0 or greater than 4.
4. Mnemonic opcode 7 should not have condition code either 0 or greater than 6.
5. Mnemonic opcode 9 and 10 should have either register or condition code part set to 0.
6. During load operation the execution time address mentioned after -1 should be equal to start address.

Assignment No. 5 – Simulator – 1

**Execution of Simple and branching based programs**

Practical Assignment

1. Implement a menu driven simulator for hypothetical Simple Instruction Computer that provides the following functionalities:
   1. Load - Loading of the program from file into memory
   2. Print - Printing the program loaded in memory
   3. Run - Executing the loaded program
   4. Exit – Exit from the program

The machine has the basic instruction set comprising of following:

|  |  |  |
| --- | --- | --- |
| Opcode | Mnemonic | Instruction |
| 00 | STOP | Stop or halt execution |
| 01 | ADD | Add memory operand to register contents |
| 02 | SUB | Subtract memory operand from register contents |
| 03 | MULT | Multiply memory operand to register contents |
| 04 | MOVER | Move memory operand contents to register contents |
| 05 | MOVEM | Move register contents to Memory operand |
| 06 | COMP | Compare register and memory operands to set condition code  appropriately |
| 07 | BC | Branch to second operand depending on condition code specified as  first operand |
| 08 | DIV | Divide register contents by memory operand |
| 09 | READ | Read into memory operand |
| 10 | PRINT | Print contents of memory Operand |

Execute sum.sm and fact.sm programs on this simulator.

Do you Know it ?

1. What is the size of memory in Hypothetical Simple Instruction Computer (SMACO)? How memory will be represented in simulation program?
2. How many registers are there in SMACO? How registers are represented in simulation program?
3. From the contents of the memory at PC, how will you separate opcode, register operand and memory operand?
4. What will happen if simulator is executing a program whose source assembly program does not contain STOP instruction?
5. What types of errors can be checked / handled by simulator.
6. Explain: absolute loader.

|  |  |  |
| --- | --- | --- |
| **Assignment Evaluation** |  | |
| 0: Not Done | 1: Incomplete | 2: Late Complete |
| 3: Needs Improvement  **Viva –Voce (0 to 2)** | 4: Complete | 5: Well Done |
| **Signature of the Teacher** | **Signature of Student** | **Date** / / |

Try it!

1. Write a SMACO program for the following:
   1. Sum of first *n* numbers. E.g. if n=5 sum=1+2+3+4+5=15
   2. Product of first *n* numbers. E.g. if n=3 Product=1x2x3=6
   3. Power of a number an
   4. GCD of two numbers
   5. Maximum of two numbers
   6. Minimum of two numbers
   7. Factorial of a number n.

Execute all programs on the above simulator.

1. Write the simulator program for a different instruction set given by your batch teacher.

Assignment No. 6 – Simulator – 2

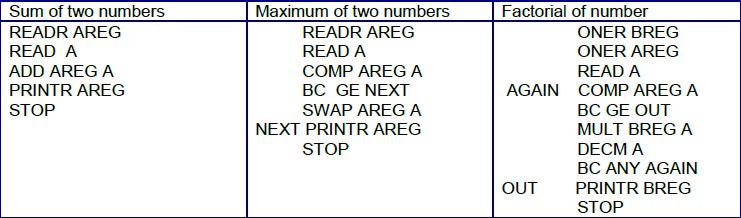
Practical Assignment

1) Extend the assignment no. 5 to incorporate following set of instruction for hypothetical Simple Instruction Computer.

The machine instruction set:

|  |  |  |
| --- | --- | --- |
| Opcode | Mnemonic | Instruction |
| 11 | SWAP | Swap the contents of memory and register operand. Both register and  memory operand |
| 12 | INCR | Increment the contents of register operand. Only register operand |
| 13 | DECR | Decrement the contents of register operand. Only register operand |
| 14 | INCM | Increment the contents of memory operand. Only memory operand |
| 15 | DECM | Decrement the contents of memory operand. Only memory operand |
| 16 | ADDM | Add the contents of register operand to memory operand. Both the  operands |
| 17 | SUBM | Subtract the contents of register operand from memory operand. Both  the operands |
| 18 | MULTM | Multiply the contents of memory operand by register operand. Both  the operands |
| 19 | DIVM | Divide the contents of memory operand by register operand. Both the  operands |
| 20 | PRINTR | Print the contents of register operand. Only register operand. |
| 21 | READR | Read into the register operand. Only register operand |
| 22 | ZEROR | Initialize register operand to zero. Only register operand. |
| 23 | ONER | Initialize register operand to one. Only register operand. |

Test the following programs on the machine given above



Do you Know it ?

1) Convert all the above programs to their respective machine language codes.

|  |  |  |
| --- | --- | --- |
| **Assignment Evaluation** |  | |
| 0: Not Done | 1: Incomplete | 2: Late Complete |
| 3: Needs Improvement  **Viva –Voce (0 to 2)** | 4: Complete | 5: Well Done |
| **Signature of the Teacher** | **Signature of Student** | **Date** / / |

Try it!

1. Write a SMAC0 program to find GCD of two numbers using the instruction set of mnemonic opcode values 0 to 23.
2. Try all the programs mentioned in the Try it! Section of assignment number 5 using the instruction set of mnemonic opcode values 0 to 23.