**COMPUTER ARCHITECTURE AND ORGANIZATION PROJECT**

[CECSC07]



**2 PASS ASSEMBLER FOR A MICROWAVE OVEN**

**BRANCH**: COE

**SECTION**: 1

Harmeet Singh (2019UCO1515)

Vinay Dahiya (2019UCO1542)

Sumit Yadav (2019UCO1548)

Tabishi Singh (2019UCO1506)

**All the assembler code files along with some documentation files can be found at** https://github.com/vinaydahiya04/Micron

**Introduction:**

In this project we designed a two-pass assembler for a primitive microwave oven, which takes in the assembly program as input and returns the resulting object code.

The Source Code For the project can be found at

**Description of the machine:**

The microwave oven under consideration has after starting takes in a temperature and time as input and then heats up the food sample at that temperature in Celsius for that time period in seconds.

The instructions are based on the Single Address Instruction format and have the accumulator as the general-purpose register.

**Architecture**:

The microwave oven comprises of 16-bit computer architecture and has a memory of 8Kb. It is made up of 12 address lines alongside 16 data lines which work in coordination for the smooth functionality of the CPU.

**Registers Used**:

This machine uses a total of 8 registers for its functioning. All these registers have been encoded using an 8 bit decimal code to identify them.

|  |  |  |  |
| --- | --- | --- | --- |
| REGISTER | ENCODING | DESCRIPTION | SIZE in bits |
| AC | 00000001 | Accumulator | 16 |
| DR | 00000010 | Data Register | 16 |
| IR | 00000100 | Instruction Register | 16 |
| AR | 00001000 | Address Register | 12 |
| PC | 00010000 | Program Counter | 12 |
| SR | 00100000 | Status Register | 8 |
| TR | 01000000 | Temperature Register | 8 |
| TMR | 10000000 | Timer Register | 16 |
| INPR | 00000011 | Input Register | 16 |

**INSTRUCTION SET:**

Memory Referenced Instructions

|  |  |  |  |
| --- | --- | --- | --- |
| **INSTRUCTION**  **SYMBOL** | **HEXADECIMAL**  **CODE** | | **DESCRIPTION** |
| **I=0** | **I=1** |
| STA X | 0XXX | 8XXX | Store value of AC directly in X or the location whose address is stored in X |
| LDA X | 1XXX | 9XXX | Load the value of X directly or indirectly into AC. |
| ISZ X | 2XXX | AXXX | Increment X and then skip the next instruction if X is zero. |
| BSA X | 3XXX | BXXX | Save return address in X and branch to the effective address of the subroutine. |
| BUN X | 4XXX | CXXX | Branch unconditionally to X. |
| OR X | 5XXX | DXXX | OR the content of AC with X and store it in AC |
| NOR X | 6XXX | EXXX | OR the content of AC with X and store it in AC |

Register Reference Instructions

|  |  |  |
| --- | --- | --- |
| **INSTRUCTION SYMBOL** | **HEXADECIMAL CODE** | **DESCRIPTION** |
| STP | 7800 | Stop the execution of the machine. |
| HLT | 7400 | Halt Timer |
| LCK | 7080 | Lock the door of the oven. |
| TMR | 7100 | Set Timer |
| ULK | 7040 | Unlock the door of the oven |
| SKZ | 7020 | Skip the next instruction if AC is Zero. |
| CLA | 7010 | Clear AC |
| CMA | 7008 | Complement AC |
| INC | 7004 | Increment AC |
| TMP | 7200 | Set Temperature of the Oven. |
| LSA | 7002 | Left Shift AC |
| ISZ | 7001 | Increment AC and skip next instruction if AC is Zero |

**Input Output Instructions:**

|  |  |  |
| --- | --- | --- |
| **INSTRUCTION**  **SYMBOL** | **HEXADECIMAL**  **CODE** | **DESCRIPTION** |
| INP | F800 | Input Character to AC. |
| OUT | F400 | Output Character from AC. |
| SKI | F200 | Skip on Input Flag. |
| SKO | F100 | Skip on Output Flag. |

**Pseudo Instructions:**

* HEX N: Hexadecimal number N to be converted to binary.
* ORG N: Execution Starts at line numbered N
* DEC N: Signed decimal number N to be converted to binary.
* END: END PROGRAM

STATUS REGISTER DESCRIPTION

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| X | X | M | H | O | I | L | P |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

P : 0-Power On

1. Power Off

L: 0-Door Unlocked

1-Door Locked

I: 0-Input not Received

1-Input Received

O: 0-Output not Received

1-Output Received

H: 0-Microwave Heater Off

1-Microwave Heater On

M: 0-Conduction Mode

1-Convection Mode

**Subroutines:**

The program has two subroutines

1) Power Subroutine

2) Heating Subroutine

The power Subroutine is used for initializing the machine setting all the flags to 0 and clearing all registers. This is mostly used at the start of the program so that the machine is ready to work properly.

POW: A00h

CLA

STA SR

STA TR

STA DR

STA TMR

STA AR

STA PC

STA IR

BUN POW I

The Heating Subroutine is the main Subroutine that handles the working of this machine. This Subroutine checks if the value of the Temperature and Timer has been received and then keeps looping through itself until the time runs out. This subroutine makes sure that the door is closed and also ensures all other requisites for proper heating effect. On the end of the time set in timer register the door is unlocked and heating is stopped.

HEA: D00 H

SKI

LDA SR

OR 10H

STA SR

LDA TMR

CMA

INC

LOP:ISZ

BUN LOP

LDA SR

OR 02H

STA SR

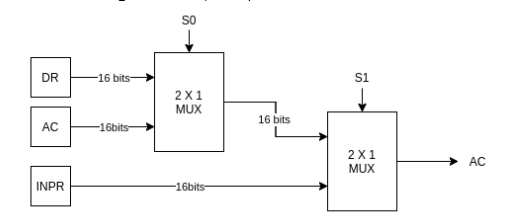
BUN HEA I

**Reserved Locations:**

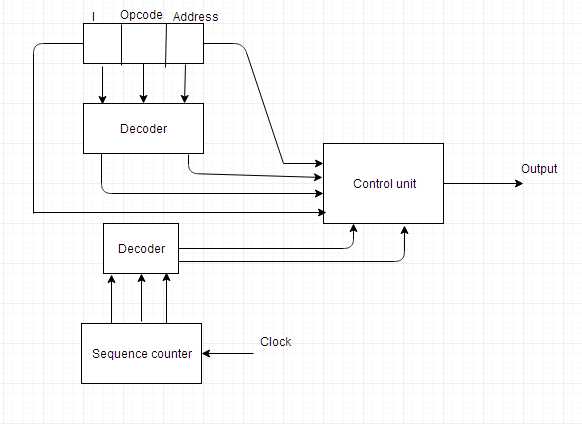
Some locations have been reserved in the main memory of the system to store the parameters of the subroutines. These memory locations should always be present as labels at the end of the assembly code.

D00H AND A00H

Also The locations ranging from 000h to 1FFh are reserved for encoding registers and other instruction referencing purposes.

**Arithmetic Logic Unit (ALU): **

**Control Unit:**



**Syntax and General Information:**

All operands, memory locations to be specified in hexadecimal form only

This machine supports single line comments only which are specified by sung the ‘$’ sign in front of the comment.

For using multi line comments use single line comments multiple times.

This machine supports four types of addressing modes:

1. Immediate

2. Direct

3. Indirect

4. Register Addressing Mode

In case of Immediate addressing mode directly specify the operand value(No more than 8bits).

In Direct and Indirect addressing mode the memory address should be specified in between square brackets [].

In Register Addressing the Registers name to be specified as operand.

The assembler is case Insensitive

**Label Format:**

[LABEL\_NAME]: {ASSEMBLY LANGUAGE INSTRUCTION}

There are certain rules defined for the declaration of the symbolic addresses and labels in your program. They are-

1. It cannot contain white spaces.
2. Cannot Start with \_
3. It can start with and contain only Alphabetic characters and \_
4. It cannot be same as a predefined label, sub-routine or opcode
5. It should contain at least 2 characters
6. It cannot be the same as an already declared  symbol / label

**Errors:**

The following points are to be kept in mind while writing the assembly code for this assembler to avoid getting any errors:

* Every program should start with an ORG N pseudo-instruction wherein N is the memory location of the word that has the first instruction.
* Apart from those mentioned above any other addressing mode is not supported.
* Each line of code must contain at most three field that are label, instruction, comments. A line of code containing any more fields is considered as an error.
* Each label should only be defined once and not doing so will result in an error.
* Label should be defined strictly adhering to the label format specified above
* All the Memory Reference Instructions must contain an operand(memory location or register) or a constant value. Failure to detect an operand in an MRI produces an error.
* All the Non-Memory Reference Instructions must not contain any operand value. An error is generated if the assembler detects an operand.
* All labels are declared using ‘:’ symbol and not using ‘,’ . Using any other symbol apart from ‘:’ will give error.

**MACHINE IS CASE INSENSITIVE**