Computer Architecture and Organization Project

Two-Pass Assembler for a Washing Machine System

Group 8:
Maanas Talwar (2019UCO1544)
Aryaman Sharma (2019UCO1508)
Anav Chaudhary (2019UCO1577)
Harshit Gupta (2019UCO1580)

Objective:

To design a washing machine controller and hence create a two-pass assembler for the same.

Note: It is assumed that the operations at the hardware level are implemented in the system itself and what we provide is a binary translated output of the provided assembly level program.

Overview:

This washing machine is a simple device consisting of two basic operations - *WASH* and *DRY*.
The system consists of timer, temperature control, spin speed and wash type controls. Subroutines are used to control these operations utilizing reserved memory locations, predefined labels and values which correspond to a particular state of temperature, spin speed and wash type.

For the working code of the 2 pass assembler please refer to the github repository.

Basic System Architecture:

The washing machine has a 16-bit computer architecture and a 12-bit, word addressable memory.
The instructions are preceded by an addressing mode bit which specifies indirect or direct addressing mode for the memory.
The instructions are based on the Single Address Instruction format and have the accumulator as the general purpose register.

INSTRUCTION FORMAT - 16 bit

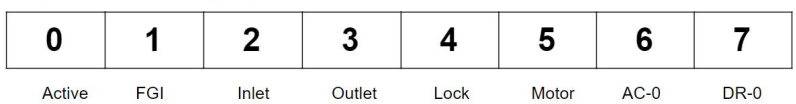
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

BIT - 0 : ADDRESSING MODE
BIT - [1,3] : OPCODE
BIT - [4,15] : MEMORY ADDRESS

Registers:

Symbol	Name	Number of Bits	Function
AC	Accumulator	16	Processor register
DR	Data register	16	Holds memory operand
PC	Program counter	12	Holds address of instruction
AR	Address register	12	Holds address for memory
IR	Instruction Register	16	Holds instruction code
INPR	Input Register	16	Holds input character
SR	Status register	8	Holds status flags

STATUS REGISTER



Instruction Set:

Memory Reference Instructions

(X ⇒ memory location or a symbolic address present as a label in the program):

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 the 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

Register Reference Instructions (take no operands)

Instruction Symbol	Hexadecimal Code	Description
SRT	7800	Start the execution of the machine
STP	7400	Stop the execution of the machine
SPN	7200	Spinning the motor of the machine before a wash cycle(sets motor flag to 1)
HLT	7100	Halt motor (sets motor flag to 0)
LCK	7000	Lock the door of the washing machine(set a flag)
ICL	7080	Clear inlet flag(inlet valve closed)
IST	7040	Set inlet flag (inlet valve open)
OCL	7020	Clear outlet flag (outlet valve closed)
OST	7010	Set outlet flag (outlet valve open) and start draining
CLA	7008	Clear AC
CMA	7004	Complement AC
INC	7002	Increment AC
SZA	7001	Skip next instruction if AC is zero

Input Output Instructions (take no operands)

Instruction Symbol	Hexadecimal Code	Description
SKI	F800	Skip if input flag is set
INP	F400	Input 8 bit information and clear input flag

Pseudo Instructions

- HEX N : Hexadecimal number N to be converted to binary
- DEC N : Signed decimal number N to be converted to binary

Subroutines:

The timing , wash-type, temperature and spin speed subroutines refer to some predefined memory locations mentioned in the Reserved Locations section. This is a software implementation of a program for a washing-machine cycle and thus the values associated with the different settings are input through external ports which indeed have representations of the values provided. The character input by the user is

transferred in the INPR register and then to the accumulator for the internal executions of these subroutines.

- TIM: Setting the timer for the machine's wash cycle only. Can accept a single integer from 1-9 minutes.
- WTP: Setting the wash type for machine cycle. Values:
  - 0 - WASH
  - 1 - DRAIN
- TMP: Setting the temperature of the water used in washing. Values:
  - 0 - COLD
  - 1 - MILD
  - 2 - WARM
  - 3 - HOT
- SSP : Setting the spin speed for the wash cycle of clothes. Values:
  - 0 : SLOW
  - 1 : NORMAL
  - 2 : FAST

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.

- hex0000: Stores the timer value input by the user. *(Label name - TM)*
- hex0001: Stores the temperature type input by the user. *(Label name - TP)*
- hex0002: Stores the spin speed type input by the user. *(Label name - SPD)*
- hex0003: Stores the wash type input by the user. *(Label name - WT)*
- hex0004: A constant value that stores the amount of time required by machine to drain and dry clothes. *(Label name - CLN)*

Register Transfer Statements:

Fetch1	T0: AR ⇐ PC
Fetch2	T1: IR ⇐ M[AR], PC ⇐ PC + 1
Decode	T2: D7…D0 ⇐ decode IR(12 - 14), AR ⇐ IR(0 - 11), I ⇐ IR(15)
Indirect	D7’IT3: AR ⇐ M[AR]

Memory Reference Instructions(SC ⇐ 0 implied at the end of all instructions):

STA	D0T4: M[AR] ⇐ AC
LDA	D1T4: DR ⇐ M[AR]
	D1T5: AC ⇐ DR
ISZ	D2T4: DR ⇐ M[AR]
	D2T5: DR ⇐ DR + 1
	D2T6: M[AR] ⇐ DR, if (DR = 0) then (PC ⇐ PC + 1)
BSA	D3T4: M[AR] ⇐ PC, PC ⇐ AR + 1
BUN	D4T4: PC ⇐ AR

Register Reference Instructions:

r ⇒ D7I'T3

B ☰ IR

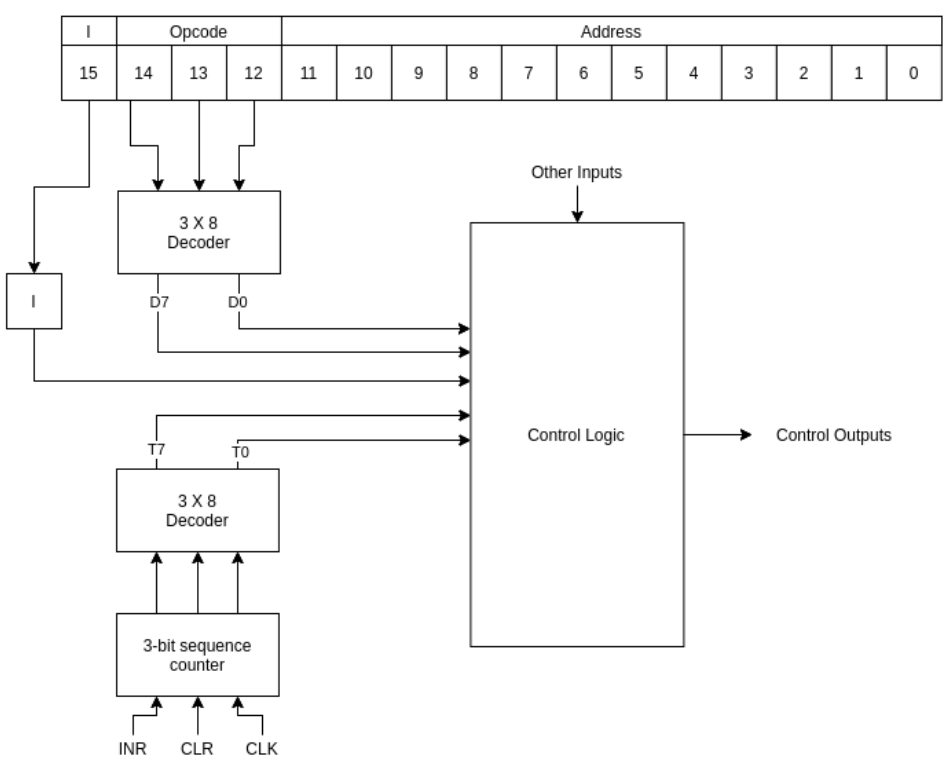
	r: SC ← 0
SRT	rB11: SR(0) ← 1
STP	rB10: SR(0) ← 0
SPN	rB9: SR(5) ← 1
HLT	rB8: SR(5) ← 0
ICL	rB7: SR(2) ← 0
IST	rB6: SR(2) ← 1
OCL	rB5: SR(3) ← 0
OST	rB4: SR(3) ← 1
CLA	rB3: AC ← 0
CMA	rB2: AC ← ~AC
INC	rB1: AC ← AC + 1
SZA	rB0: if (AC = 0), then PC ← PC + 1
LCK	r~(B0+B1+B2+B3+B4+B5+B6+B7+B8+B9+B10+B11): SR(4) ← 1

Input Output Reference Instructions:

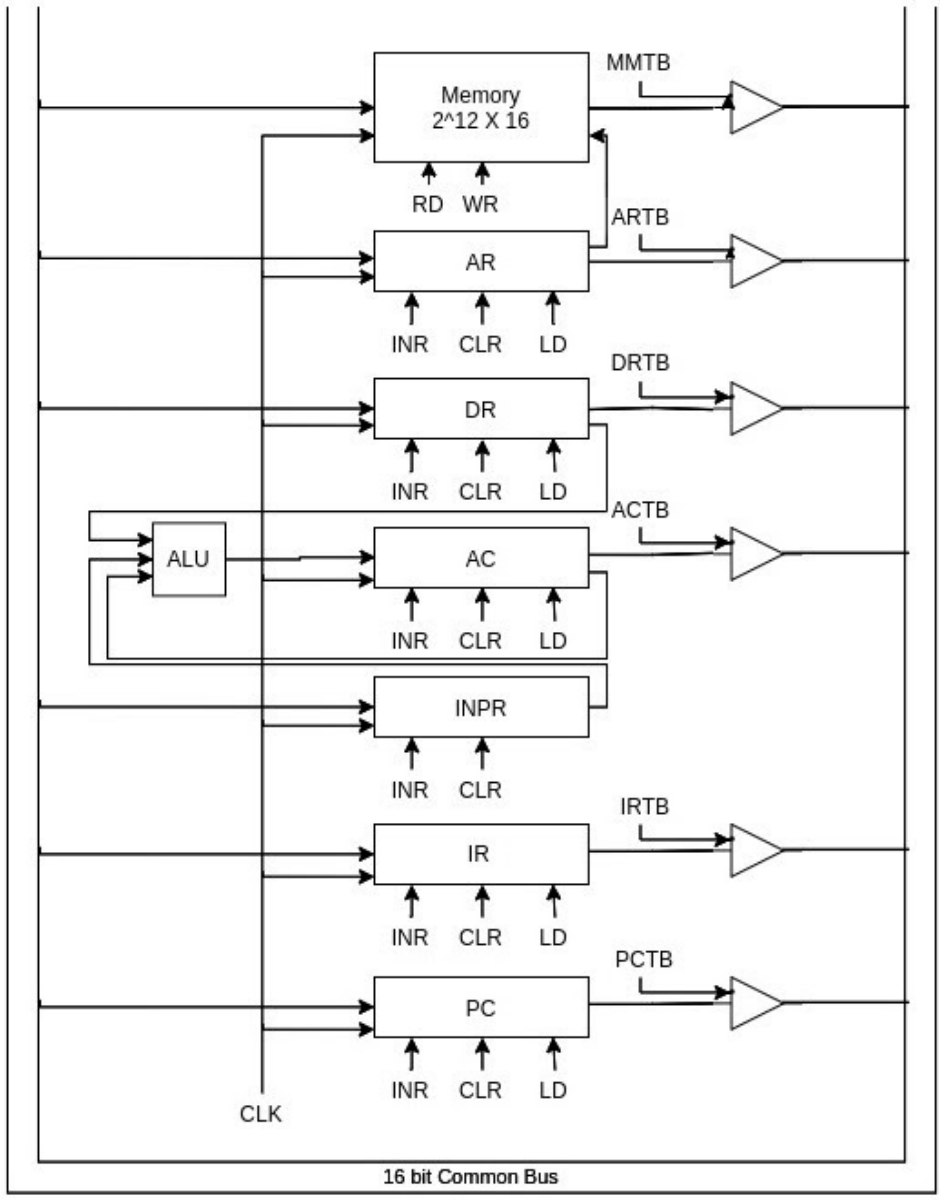
p ⇒ D7IT3

SKI	pB11: if(SR(1) = 1), then (PC ← PC + 1)
INP	pB10: AC ← INPR, SR(1) ← 0

Control Unit:

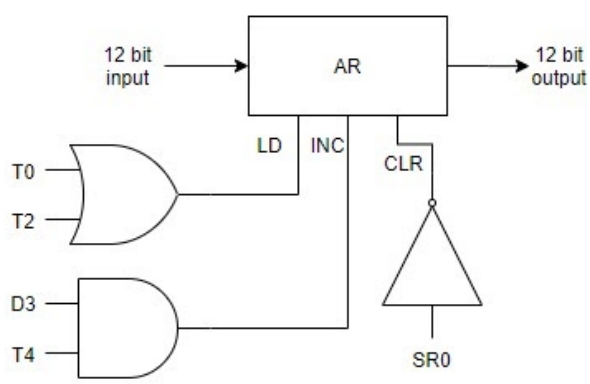


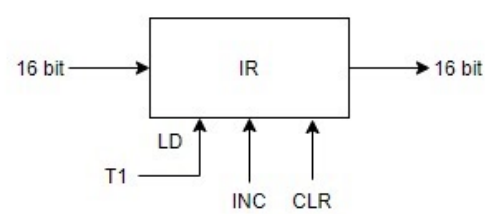
Common Bus Architecture:

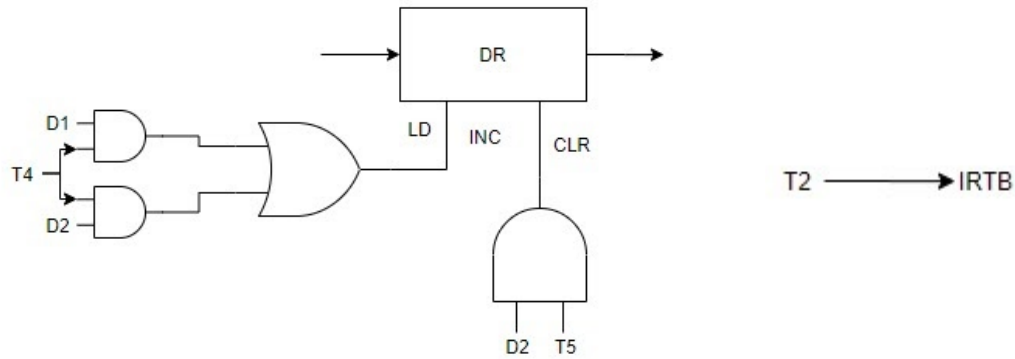


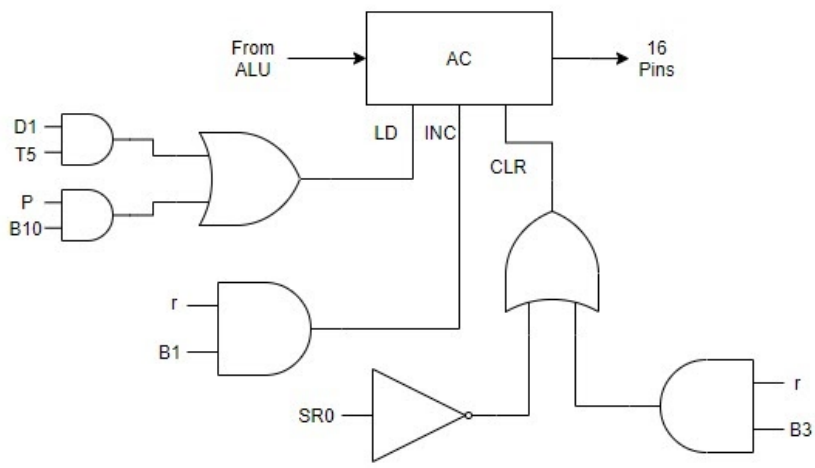
Control Logic:

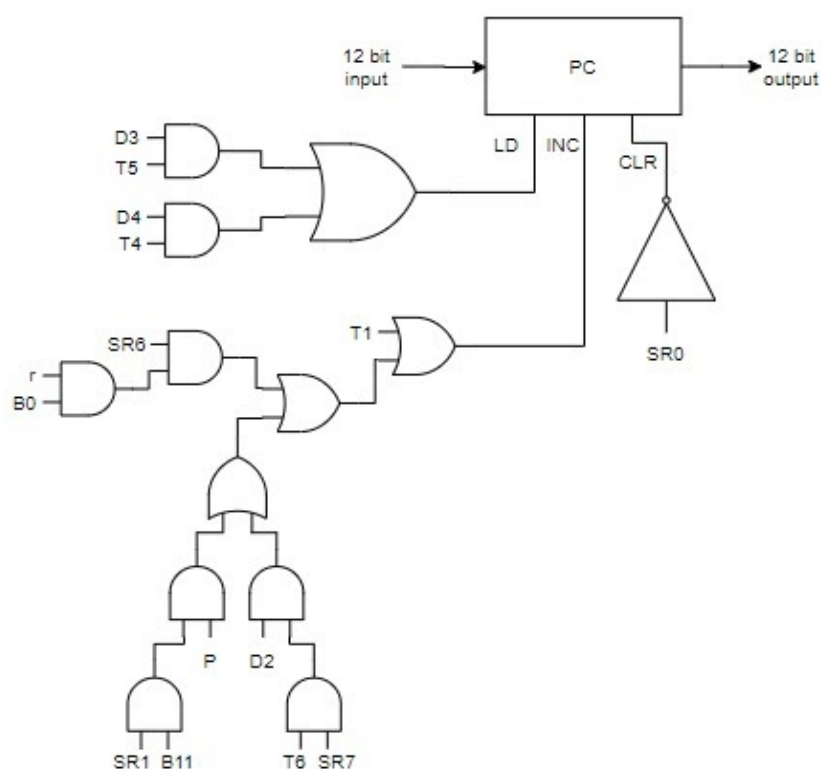
Register and Memory Controls:

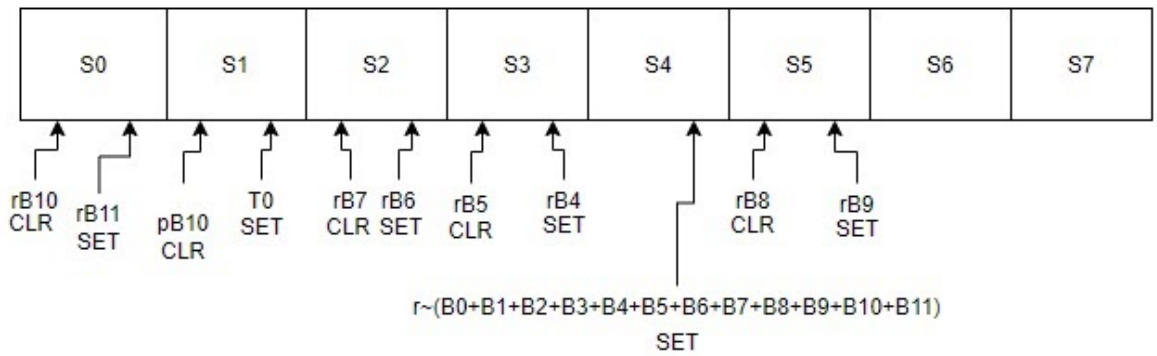


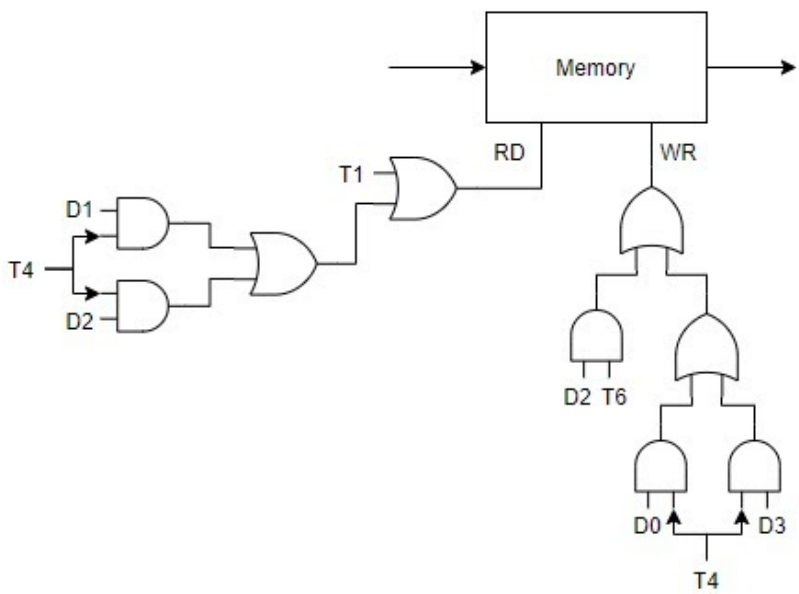



DR
D1
T4
D2
LD
INC
CLR
D2
T5
T2
IRTB

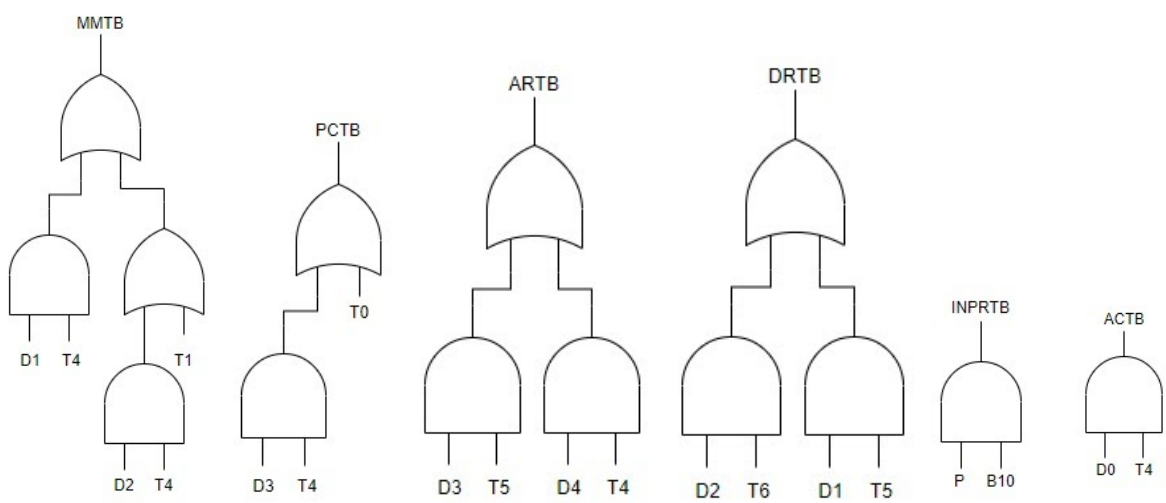

From
ALU
AC
16
Pins
D1
T5
P
B10
LD
INC
CLR
r
B1
SR0
r
B3


12 bit
input
PC
12 bit
output
D3
T5
D4
T4
LD
INC
CLR
SR0
T1
SR6
r
B0
P
D2
SR1
B11
T6
SR7

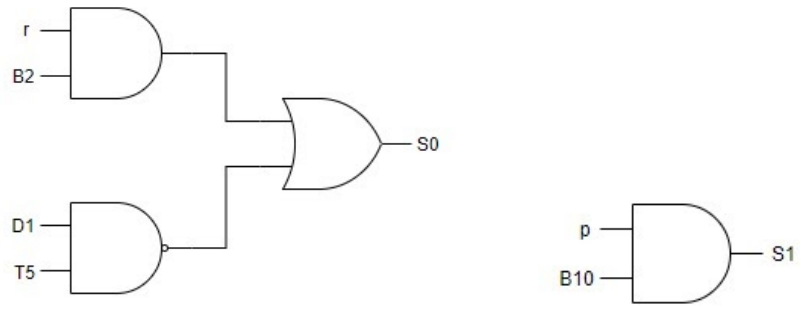




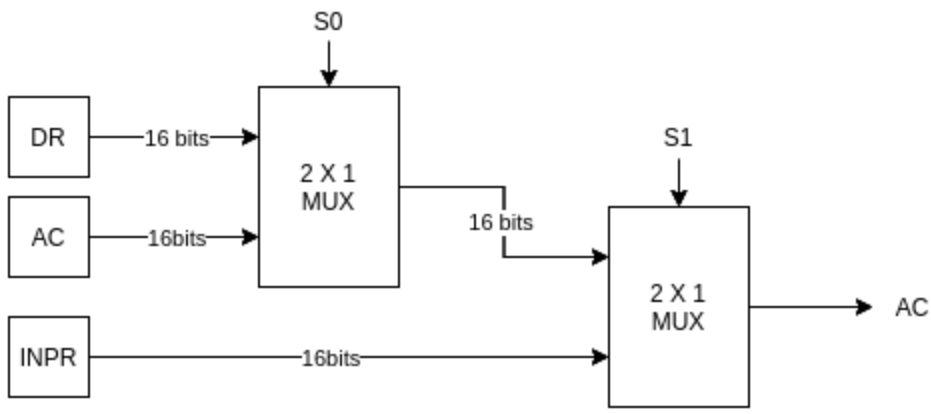
Bus Controls:



Arithmetic and Logic Controls:



Arithmetic and Logical Unit(ALU):



16 such units will operate on each bit of the 16 bit registers i.e. DR, AC and INPR.

S1	S0	Selected
0	0	DR
0	1	AC
1	X	INPR

Assembly Language Syntax:

Any variable or symbolic address referred to in the following lines is enclosed in '[]' and all comments are preceded by ';'. Instruction codes are written as they are.

- Each line of code consists of three fields - label, instruction and comment. Each line of code is terminated by a new line.
- Every program should start with SRT operation which denotes the start of execution of the machine, failing which generates an error.
- The program may include comments which are preceded by a ';' sign.
- Every program must contain an STP instruction at the end of the assembly program, failing which generates an error. However, lines of code may be present after the STP instruction so as to define symbols.

Example:

1. ... ; Program
   STP
2. ... ; Program
   STP
   A, hex 1000
   B, hex 2000

Opcodes, Operands and Addressing:

- All the opcodes and operands are separated by one or more spaces. All the operands are case sensitive while opcodes are not.
- The Memory Reference Instructions occupy two or three symbols separated by spaces:
    a. 3-letter symbol defining an MRI
    b. Symbolic address/Operand
    c. letter I, after the operand field, to denote indirect addressing. Direct Addressing is assumed for any MRI instruction not ending in I.
- A Non-MRI must only contain a 3 - letter symbol specifying one of the valid Non-MRI instructions.
- Every line should be terminated by a new line at the end.

Syntax

opcode [operand_name] [I]

Examples:

LDA and lda are the same opcodes.

Var, VAR, var are different operands

```
CLA       ; non-MRI
LDA var I ; Indirect Addressing MRI
LDA var   ; Direct Addressing MRI
```

Symbolic addresses and Labels:

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

1. It can not contain spaces
2. It may not start with a numeric value
3. It can start with and contain only Alpha-numeric characters and _
4. It can not be same as a predefined label, sub-routine or opcode
5. It can not be the same as an already declared symbol / label
6. It must contain at most 3 characters

Examples:

<u>Valid symbolic addresses and labels</u>

VAR

a
AB_
t1

Invalid symbolic addresses and labels
2va ; starts with a numeric value
t.v ; contains "." character
m v ; contains a space in the declaration
SPN/spn ; same as a predefined instruction
name ; contains more than 3 letters

Syntax for labels:

- The label name must be followed by a ":" immediately after it and the code associated with the label may start from the same line,separated by one or more spaces, or from the next line.
- If the label defines an operand, it must be followed by a "," without a space. The value associated with the symbol must be defined in the same line after the ",".

[LABEL_NAME]: additional code
[LABEL_NAME], value

Errors:

The following rules have to be taken care of while writing a program for the execution of a washing machine assembly program:

- Every program must contain an SRT (START) instruction as the first instruction. Failure to do so would result in a fatal error and the execution of the assembler would not be initiated.
- Each and every line must be terminated by a new line. Not doing so results in a syntax error.
- Each line of code must contain at most three fields that are label, instruction, comments. A line of code containing any more fields is considered as an error.
- Each symbolic address that is mentioned in any instruction field must occur again as a label field. If not found, an error is generated.
- No symbolic address or label must be the same as a predefined opcode, sub-routine or label field. If a same name is found, an error is generated.
- All the Memory Reference Instructions must contain an operand 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.

For the working code of the 2 pass assembler please refer to the github repository.