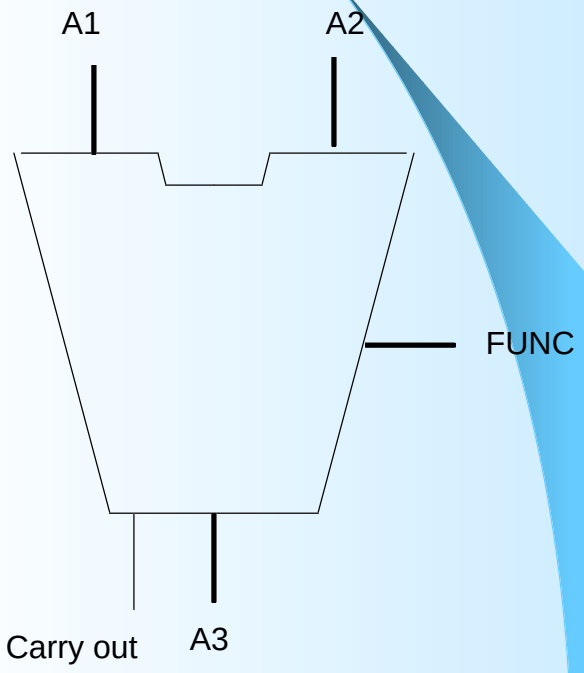
CPU Architecture

- This is just a refresh on the internal workings of a simple CPU.
- CPU has:
  - 8 bit data bus
  - 16 bit address bus

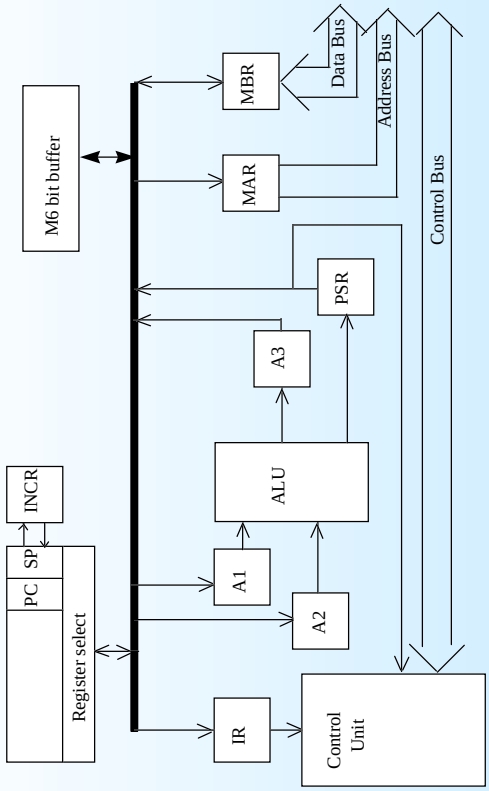
The ALU



Function of the ALU

- Depending on the value of the Function code:
  - Add
  - Subtract
  - AND
  - OR
  - Complement
  - possibly
    - multiply
    - divide

Complete internals of a simple CPU



Register Descriptions

The following elements are all some form of register made up from flip-flops.

- A1, A2, A3
- Data can be loaded into the registers A1 and A2 via the bus.
- Data can be read from A3 via the bus

MBR

- MEMORY BUFFER REGISTER OR MEMORY DATA REGISTER.
  - Data can be transferred to or from this register via the internal bus.
  - externally connected to the data bus

MAR

- MEMORY ADDRESS REGISTER.
  - Data can be loaded into this register via the internal bus

16 bit BUFFER

- Used for storing the 16 bit address operand
  - The most significant byte is read from the Data bus and stored in this top half of this register
  - then the least significant byte is read from the Data bus and stored in the lower half of this register.
  - The contents are transferred when required to the MAR.

PSR

- PROGRAM STATUS REGISTER.
  - Indicates the current value contained in the ALU
    - = zero
    - > zero
    - < zero
    - carry

IR

- INSTRUCTION REGISTER.
  - Contains the current instruction being executed by the Control Unit

SP

- STACK POINTER.
- A counter which can be incremented or decremented.
  - Used for Stack manipulations.

PC

- PROGRAM COUNTER.
- A Counter which is used to point to the next instruction
  - automatically incremented after each access.
  - can be set to any value via the internal bus.

THE CONTROL UNIT.

- FUNCTION OF THE CONTROL UNIT.
  - To implement the current instruction in the IR
  - To be in control of all the register to register transfers.
  - To respond to the output from the PSR.
  - To monitor the external Control bus inputs and drive the Control bus outputs.

TYPICAL INSTRUCTIONS

- Some common instruction types are:
  - Access data
  - Manipulate data
  - Store data

Examples:

- Read a location and store in a register.
- Add the contents of a location and a register.
- Write the contents of a register to a location.

Instruction Sets

- A processor recognises each instruction as an individual byte ***(the opcode)***

plus any associated data ***(the operand***)

Example:

- Read and store the value in Reg A1 of the contents of address C001h

Instruction Format

Opcode	Operand	
Description of what we want to happen		
Read and store the value in Reg A1, of location	C001h	
Code that the CPU requires (as bytes)		
A6	C0	01

Instruction Cycle

- Fetch instruction
  - Transfer PC to MAR
    - PC incremented
    - Read data into MBR
  - Transfer MBR to IR
- Control Unit now knows what the Instruction is.
  - Read operand (first byte)
    - Transfer PC to MAR (PC incremented)
      - Read MBR (msb of Address)
      - Transfer to msb of 16 bit buffer

Instruction Cycle (cont)

  - Read operand (second byte)
    - Transfer PC to MAR (PC incremented)
      - Read MBR (lsb of Address)
      - Transfer to lsb of 16 bit buffer
- Read location
  - Transfer 16 bit buffer to MAR
    - MAR = 0C001H
    - Read value on MBR
    - Transfer to A1
- Instruction complete.

Exercise

- Explain the interaction required and the values transferred, between the internal registers to perform the following instruction:
  - Add the contents of Register A1 and the contents of memory location 3000H, the result **will be** in register A3.
    - You may assume that the PC contains the address of the opcode for this instruction.
    - A1=55H and 3000H contains 0AH

Fundamental Rules of Circuit Design.

- Very similar to rules governing software design.
- Well structured
- Good documentation
- Clear code
- Make everything obvious.
- Do not revert to Assembly level programming if at all possible.

The Algorithmic State Machine (ASM)

- Most digital systems require that signals are sequenced through a set of different processes to provide the end result. Compare to the similar problems in writing code.

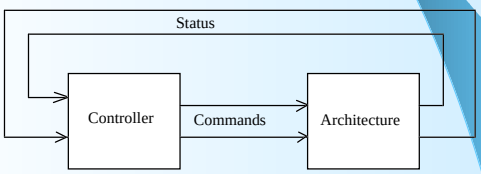
Design Style.

- Design from top down.
- Maintain clear distinction between controller and the controlled hardware.
- Develop a clearly defined architecture and control algorithm before making detailed decisions about hardware.

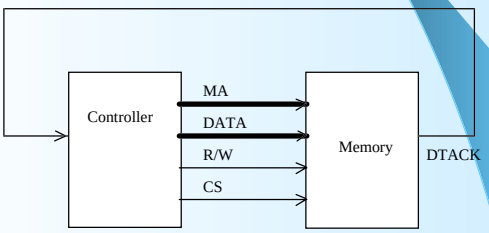
Design from top down.

- Is the problem clearly stated?
- Could the problem be restated more clearly or more simply?
- If working on a sub-system of a larger design do examine the overall system.

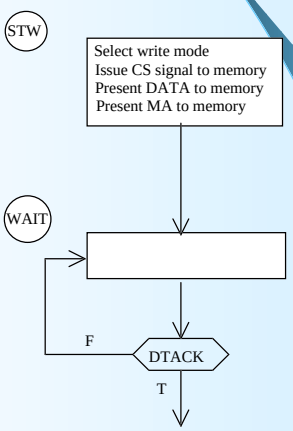
Maintain clear distinction between controller and the controlled hardware.



Taking the example of writing data to memory.



Control Algorithm.

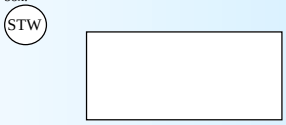


ASM Chart Notations.

ASM Chart Notations.

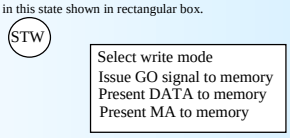
States

Name inserted in circular box.

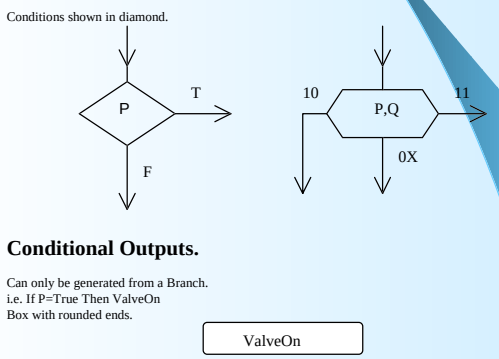


Outputs.

All outputs manipulated in this state shown in rectangular box.



Branches and Conditional Outputs.



Conditional Outputs.

Can only be generated from a Branch.
i.e. If P=True Then ValveOn
Box with rounded ends.

ValveOn

The Washing Machine.

- As an example we shall design a simple washing machine. To fulfil the following specification:
- Wash time programmable between 1 to 30 minutes.
- If door opened machine must stop until door closed then continue.
- Drum reverses direction every 30 seconds.

Sensors/Inputs.

- SwOn.H, starts the washing cycle.
- LevelOK.H indicates when the water has reached the correct level in the machine.
- DoorShut.H indicates that the loading door is closed.
- 6 bit counter incremented every 30 seconds.
- Time select 1 through to 32 minutes gives a 6 bit output pattern.

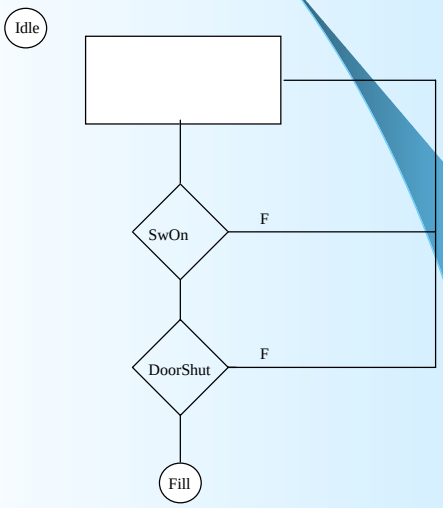
Outputs

- MotorON.H energises the motor
- Clockwise (CW.H) determines the direction of rotation of the motor.
- WaterON.H energises a solenoid to allow water into the machine.

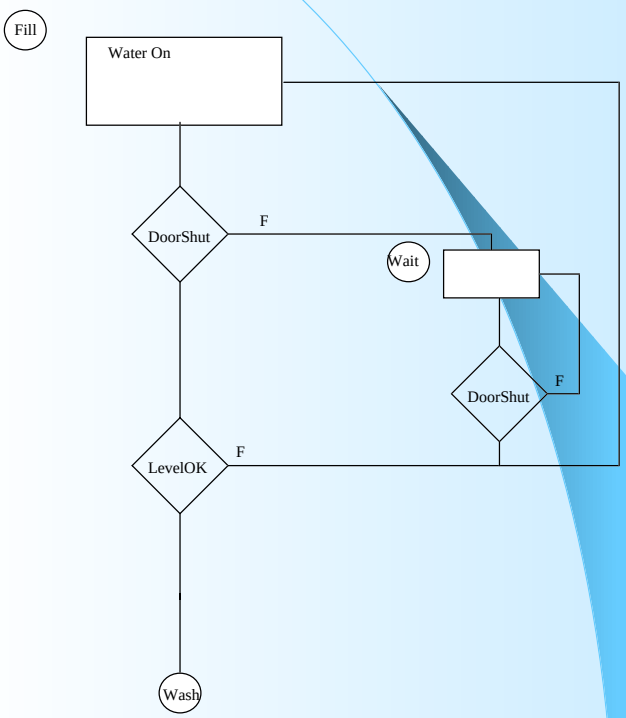
ASM for the machine.

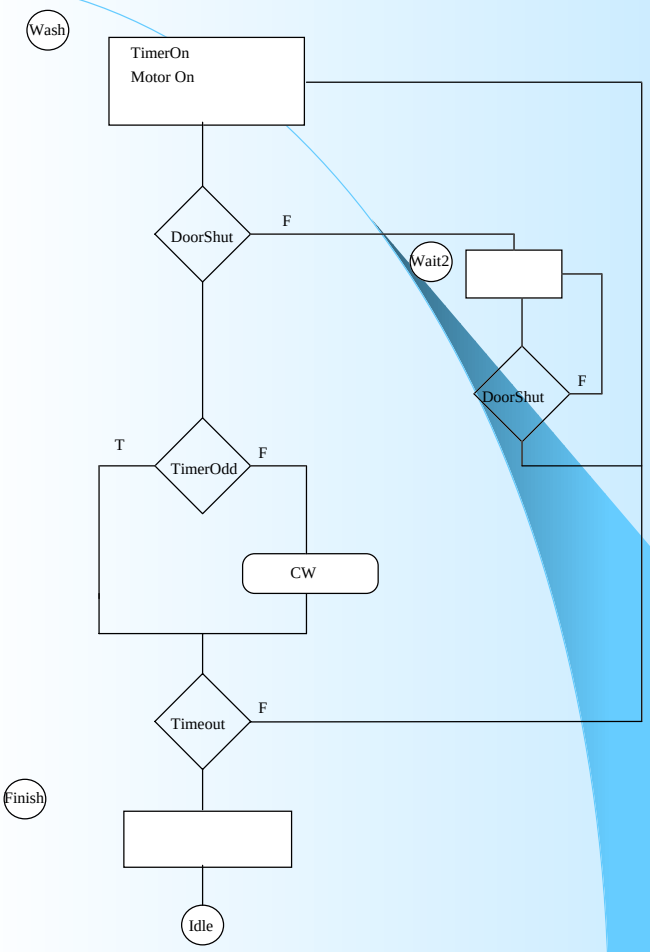
- What has the machine to do?
- Perform a wash cycle which could be broken down into:
  - Waiting for go (Idle)
  - Filling
  - Washing
  - Finished

The Idle State.



The Fill State





State Transition Data

State Name	Next State Name	Condition for transition
Idle	Idle	/SwOn + /Door Shut
	Fill	SwOn.DoorShut
Fill	Fill	DoorShut./LevelOK
	Wait	/DoorShut
	Wash	DoorShut.LevelOK
Wait	Wait	/DoorShut
	Fill	DoorShut
Wash	Wash	DoorShut./Timeout
	Wait2	/DoorShut
	Finish	Timeout
Finish	Idle	
Wait2	Wait2	/DoorShut
	Wash	DoorShut