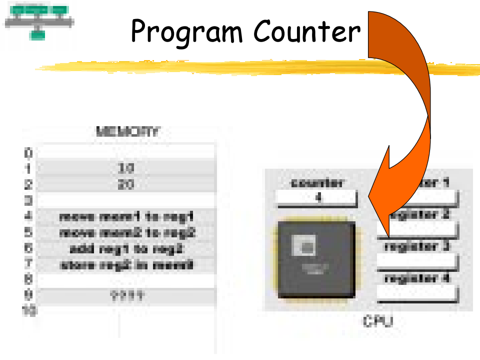
Program Counter

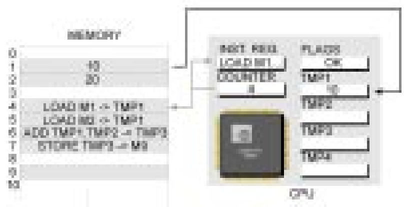
- A **Program Counter** (**PC**) is a register which always holds the memory location of the **next** instruction to be executed.
- The CPU reads the instruction in the memory location specified by the PC.
- It decodes this instruction to determine exactly the operation to be carried out.
- The CPU starts at this point, steps through each instruction until told to stop.
- At this point this point the monitor resumes execution.

Program Counter



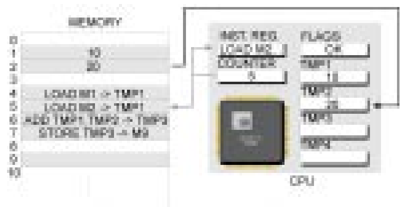
Machine Code Execution

1st CYCLE



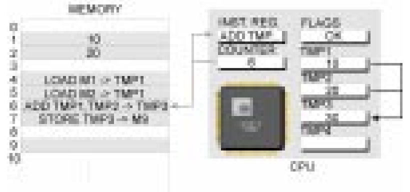
Machine Code Execution

2nd CYCLE



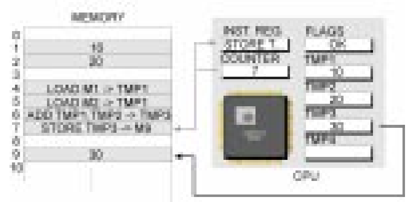
Machine Code Execution

3rd CYCLE



Machine Code Execution

4th CYCLE



The Fetch-Decode-Execute Cycle Revisited

- Programs consist of sequences of instruction found in consecutive memeory locations
- Instructions occupy 1 to 5 words of memory
- Example:
  - **rts** (Return from Subroutine) = 1 word
  - **move #5,D1** = 2 words

More Fetch-Decode-Execute Cycle

When assembled, instructions = 1 **OpCode** (**Op**eration **Code**) + 0 or more Operations:

```
move #5,D1   →   323c 0005
```

OpCode	Operand #1
323C	0005

More Fetch-Decode-Execute Cycle

```
move #5,D1   →   323c 0005
```

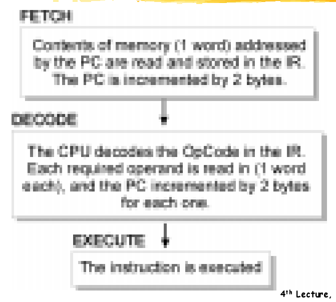
OpCode	Operand #1
323C	0005

- The OpCode specifies the type of instruction (e.g. Move data to register D1)
- Operands are what the instruction opeartes on. (e.g. value 5)
- -> This instruction loads the value 5 into the CPU register D1

How to keep track?

- How does the CPU keep track of where it is, if an instruction has a number of operants?
  - When instructions are decoded the CPU determines the no. of operands (according to the OpCode).
  - The PC is incremented by 2 bytes for each operand word.

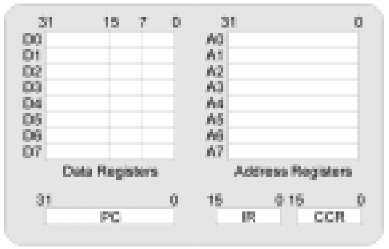
Fetch-Decode-Execute


FETCH
Contents of memory (1 word) addressed by the PC are read and stored in the IR. The PC is incremented by 2 bytes.
DECODE
The CPU decodes the OpCode in the IR. Each required operand is read in (1 word each), and the PC incremented by 2 bytes for each one.
EXECUTE
The instruction is executed

Simple Overview of the MC6800

- The MC6800/MC68332 CPU has:
  - 8 x 32-bit Data Register, D1 to D7
  - 8 x 32-bit Address Register, A1 to A7
  - 1 x 32-bit Program Counter, PC
  - 1 x 16-bit Instruction Register, IR
  - 1 x 16-bit Status Register, SR
- During one memory access the CPU can access 1 word (16-bit Data Bus). The PC is 32 bits wide, but omly 24 bits are used on the Robot (24-bit Address Bus).

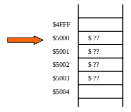
The MC6800 CPU



Simplified MC68000 CPU Block Diagram

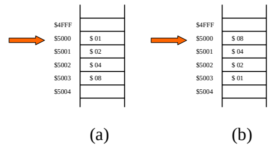
How is data stored?

- Consider the instruction:
  move.l #$01020408, $5000
- Starting @ location $5000 write out the values of the 4 occupied bytes



How is data stored?

- Consider the instruction:
  move.l #$01020408, $5000



Endianess

- Multi-byte data types (word, long-w.) are stored differently on different architectures
  - Big-endian architecture stores the lower bytes (lower addresses) are the most significant bytes
    - M68000/VAX use this approach
  - Little-endian architecture stores the higher bytes (higher addresses) as the most significant
    - x86 use this approach
  - Others but don't bother

Endianess(2)

- Example consider the instruction
  - move.l #$01020408, $5000
  - Look at memory starting @ location $5000

	Big	Little
$5000	$01	$08
$5001	$02	$04
$5002	$04	$02
$5003	$08	$01

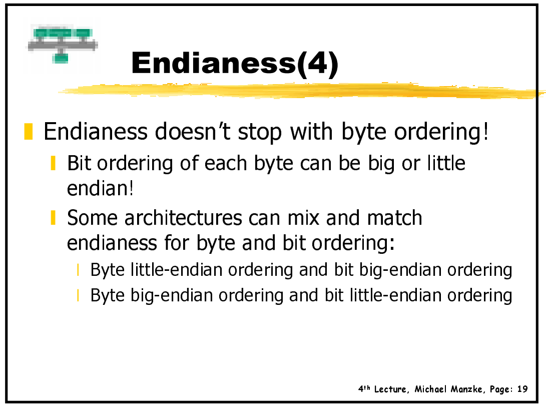
…….

Endianess(3)

- Example consider the instruction
  - move.w #$0102, $4000
    move.w #$0408, $4002
  - Look at memory starting @ location $4000

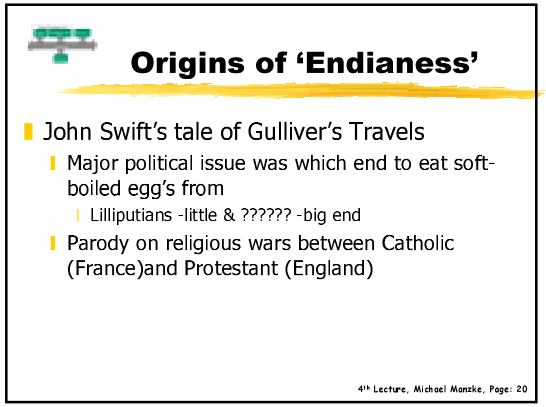
	Big	Little
$4000	$0102	$0201
$4002	$0408	$0804

…….



Endianess(4)

- Endianess doesn't stop with byte ordering!
  - Bit ordering of each byte can be big or little endian!
  - Some architectures can mix and match endianess for byte and bit ordering:
    - Byte little-endian ordering and bit big-endian ordering
    - Byte big-endian ordering and bit little-endian ordering



Origins of 'Endianess'

- John Swift's tale of Gulliver's Travels
  - Major political issue was which end to eat soft-boiled egg's from
    - Lilliputians -little & ?????? -big end
  - Parody on religious wars between Catholic (France)and Protestant (England)