Multicycle implementation of mMIPS

- Why multicycle?
- Drawbacks of single cycle datapath implementation of mMIPS
- A low latency instruction such as "Jmp"
  - Would've computed the result early during clock cycle
  - But long wait until end of clock cycle
- Natural improvisation
- Discretize time into smaller duration clock cycles
- Faster frequency clock signal
- Different instructions would require different number of such short clock cycles
  - Eg. "load word" — 5 clock cycles
    " jmp" — 2 clock cycles
Edge-triggered synchronous sequential ckt to implement MIPS

- talking in terms of clock cycles

- updates of 'state' of computation

only happening at the end of the clock cycle
(at the triggering edge of clock)

- 'state of computation' is stored in registers, memory blocks...
FSM + datapath

- Multi-cycle MM/PS
- Case study of FSM + Datapath

Instruction cycle: sequence of clock cycles during which an instruction is getting processed.

Instruction subcycle: individual clock cycles of instruction cycle.
Instruction-cycle

- suboperations
  - Fetch
  - Decoding & reading operands from Reg File
  - Arith & logic execution
  - Memory access
  - Write Back
FSM diagram

- **Fetch**
  - **Decode**
    - **Ex**
      - **WB**

Actions:
- **op != jmp**
- **op == branch**
- **(op != branch) && (op != ALU type)**
Datapath for multicycle implementation of MIPS architecture components:

- PC
- ALU
- Memory (we'll see that single memory for I+D)
- Reg File

? what more? - for storing results of suboperations
More registers for multicycle matters:

- **IR**: for storing/registering the result of Fetch subcycle
  
- **MDR**: (memory data register) storing result of suboperation MemAccess during instruction cycle of "load word" instruction
A register

B register

ALU result

ALU result

To store result of Ex suboperation
a mux required

used for WB

WB state of "load word"

used for WB

of say "add" instr cycle