



## Memory

- ❑ From outside memory is 256 words of 8-bits each
  - Separate *writedata* and *memdata* ports
- ❑ Internally 64 words of 32-bits each
  - Upper 6 bits of *adr* used to select which word
  - Lower 2 bits of *adr* used to select which byte
- ❑ At initialization, loaded from a file named "memfile.dat"
  - Whose format is as a ".csv" like file
  - Where each line in file is contents of a 32-bit word
  - And each word expressed as 8 hexadecimal digits
  - With the 1<sup>st</sup> word going into word[0], the next into word[1], etc
    - You do not need to load the whole memory
- ❑ During operation, it is always "reading" to *memdata*
- ❑ Write operation occurs "at" rising edge of clock
  - *adr* and *writedata* presented at same time as *memwrite* goes to 1

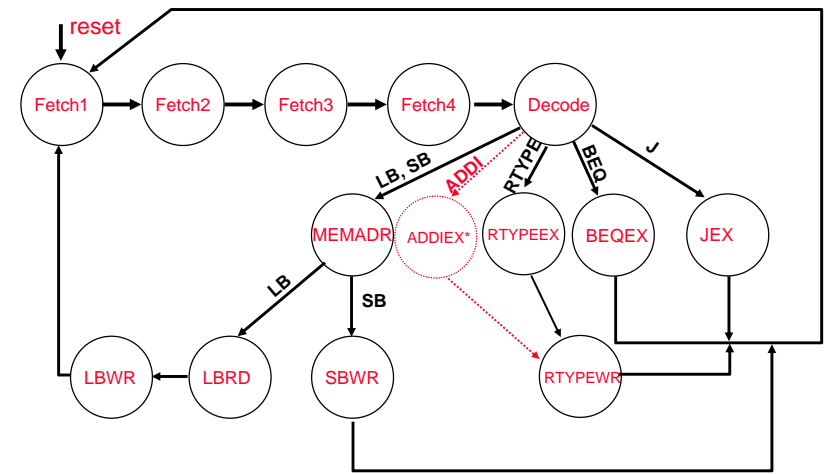
## Top module (similar to "testbench")

- ❑ Instantiates the *mips* core and the *exmemory*, and interconnects them
- ❑ Starts with raising *reset* to 1 for 22 time units, then dropping it
- ❑ Also generates a clock of 10 time unit period
- ❑ Also includes a load program specific termination test:
  - If the program ever writes to location 5
    - And the data is a "7", then success
    - Else failure
- ❑ Writing from *writedata* into the memory occurs on the rising edge of the clock

## Controller module (Behavioral)

- ❑ States
  - FETCH1, FETCH2, FETC3, FETCH4: 4 states to read 32b instruction
  - DECODE: decode just fetched instruction
  - MEMADR: computes a memory address
  - RTYPEEX: execute R-type opcode
  - RTYPEWR: write result back at end of R-type opcode into reg file
  - LBRD: read data from memory into core
  - LBWR: write data just read from memory into reg file
  - SBWR: write data to memory
  - BEQEX, JEX: execute states for BEQ or J opcodes
  - **ADDIEX: new state for ADDI implementation**
- ❑ Reset changes state to *FETCH1* state
- ❑ Internal state changes on rising edge of clock
- ❑ Control signals assume their values starting at rising clock

## State Diagram



\* added for ADDI implementation

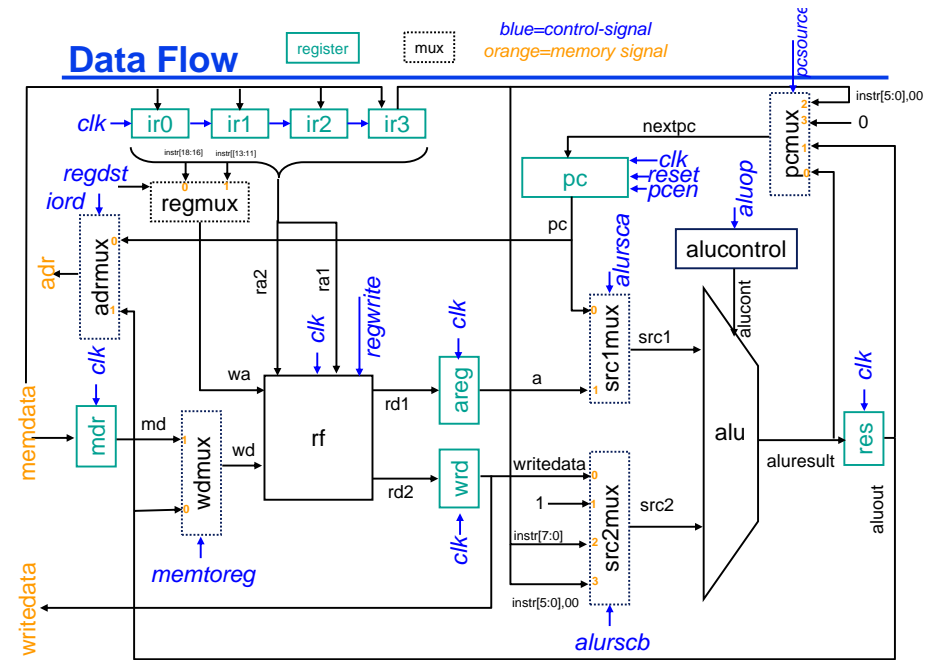
## Instruction Cycle Table

Opcode	# Cycles	Cycles (Starting with DECODE)
ADD	7	DECODE, RTYPEX,RTYPEWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
ADDI	7	DECODE, ADDIEX,RTYPEWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
AND	7	DECODE, RTYPEX,RTYPEWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
BEQ	6	DECODE, BEQEX,IFETCH1,IFETCH2,IFETCH3,IFETCH4
J	6	DECODE, BEQEX,IFETCH1,IFETCH2,IFETCH3,IFETCH4
LB	8	DECODE, MEMADR,LBRD,LBWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
OR	7	DECODE, RTYPEX,RTYPEWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
SB	7	DECODE, MEMADR,SBWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
SLT	7	DECODE, RTYPEX,RTYPEWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4
SUB	7	DECODE, RTYPEX,RTYPEWR, IFETCH1,IFETCH2,IFETCH3,IFETCH4

CSE 462 mips-verilog.9

Kogge, ND, 9/24/09 3/7/08

## Data Flow



CSE 462 mips-verilog.10

Kogge, ND, 9/24/09 3/7/08

## Register File module

- 2 read, 1 write port
- Always reading on read ports
  - I.e. change the register address on *ra1* or *ra2* and *rd1*, *rd2* change immediately
- Writing occurs at rising edge of clock
  - if *regwrite* signal is active

## Datapath Module (Structural)

- Instruction register implemented as 4 8-bit latches
  - *ir0*, ... *ir3*
  - Loaded sequentially during IFETCH
- *pcreg*:
  - Reset to zero on a *reset* high
  - Loaded from *pcmux*
  - ALU used to increment *pc*
- Includes internal staging latches (store on rising edge)
  - *areg*: capture output of read port 1 of reg file
  - *wrd*: capture output of read port 2 of reg file
  - *res*: capture output of alu
  - *mdr*: capture read data output from memory