CS 61C:
Great Ideas in Computer Architecture
Single Cycle MIPS CPU—Part I

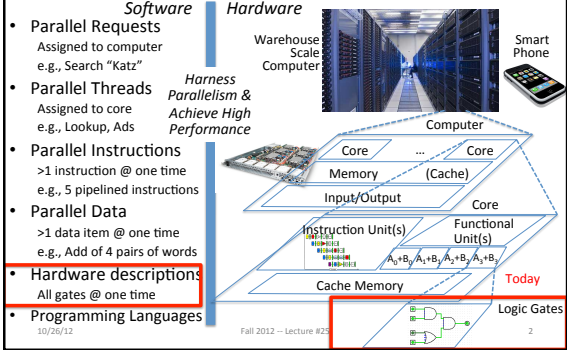
Instructors:
Krste Asanovic, Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/fa12

You are Here!

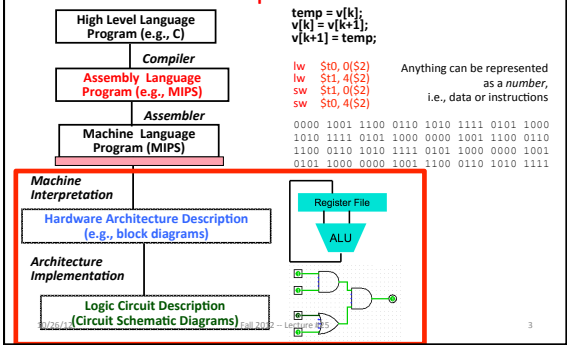
Software | *Hardware*

- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance



Levels of Representation/ Interpretation



Review

- Hardware systems made from *Stateless* Combinational Logic and *Stateful* "Memory" Logic (Registers)
- Clocks tell us when D-flip-flops change
  - Setup and Hold times important
- We pipeline long-delay CL for faster clock cycle
  - Split up the *critical path*
- Finite State Machines extremely useful
- Use muxes to select among input
  - S input bits selects 2^S inputs
  - Each input can be n-bits wide, indep of S
- Can implement muxes hierarchically
- Can implement FSM with register + logic

Agenda

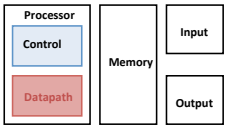
- MIPS-lite Datapath
- Administrivia
- CPU Timing
- MIPS-lite Control

Agenda

- MIPS-lite Datapath
- Administrivia
- CPU Timing
- MIPS-lite Control

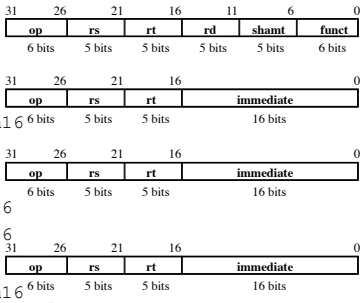
Processor Design Process

- Five steps to design a processor:
  1. Analyze instruction set → datapath requirements
  2. Select set of datapath components & establish clock methodology
  3. Assemble datapath meeting the requirements
  4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  5. Assemble the control logic
     - Formulate Logic Equations
     - Design Circuits



The MIPS-lite Subset

- ADDU and SUBU
  - `addu rd,rs,rt`
  - `subu rd,rs,rt`
- OR Immediate:
  - `ori rt,rs,imm16`
- LOAD and STORE Word
  - `lw rt,rs,imm16`
  - `sw rt,rs,imm16`
- BRANCH:
  - `beq rs,rt,imm16`



Register Transfer Language (RTL)

- RTL gives the meaning of the instructions

```
{op , rs , rt , rd , shamt , funct} ← MEM[ PC ]
{op , rs , rt ,   Imm16} ← MEM[ PC ]
```

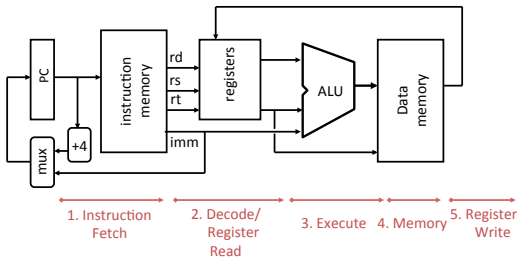
- All start by fetching the instruction

```
Inst   Register Transfers
ADDU   R[rd] ← R[rs] + R[rt]; PC ← PC + 4
SUBU   R[rd] ← R[rs] – R[rt]; PC ← PC + 4
ORI    R[rt] ← R[rs] | zero_ext(Imm16); PC ← PC + 4
LOAD   R[rt] ← MEM[ R[rs] + sign_ext(Imm16)]; PC ← PC + 4
STORE  MEM[ R[rs] + sign_ext(Imm16) ] ← R[rt]; PC ← PC + 4
BEQ    if ( R[rs] == R[rt] )
           then PC ← PC + 4 + (sign_ext(Imm16) || 00)
           else PC ← PC + 4
```


Step 1: Requirements of the Instruction Set

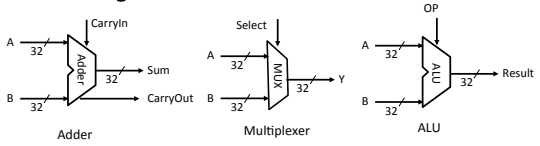
- Memory (MEM)
  - Instructions & data (will use one for each: really caches)
- Registers (R: 32 x 32)
  - Read *rs*
  - Read *rt*
  - Write *rt* or *rd*
- PC
- Extender (sign/zero extend)
- Add/Sub/OR unit for operation on register(s) or extended immediate
- Add 4 (+ maybe extended immediate) to PC
- Compare if registers equal?

Generic Steps of Datapath



Step 2: Components of the Datapath

- Combinational Elements
- State Elements + Clocking Methodology
- Building Blocks



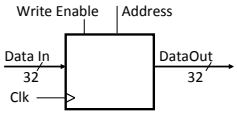
ALU Needs for MIPS-lite + Rest of MIPS

- Addition, subtraction, logical OR, ==:

```
ADDU   R[rd] = R[rs] + R[rt]; ...
SUBU   R[rd] = R[rs] - R[rt]; ...
ORI    R[rt] = R[rs] | zero_ext(Imm16)...
BEQ    if ( R[rs] == R[rt] )...
```

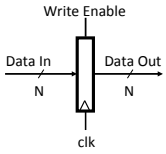
- Test to see if output == 0 for any ALU operation gives == test. How?
- P&H also adds AND, Set Less Than (1 if A < B, 0 otherwise)
- ALU from Appendix C, section C.5

Storage Element: Idealized Memory



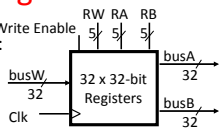
- Memory (idealized)
  - One input bus: Data In
  - One output bus: Data Out
- Memory word is found by:
  - Address selects the word to put on Data Out
  - Write Enable = 1: address selects the memory word to be written via the Data In bus
- Clock input (CLK)
  - CLK input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block: Address valid ⇒ Data Out valid after "access time"

Storage Element: Register (Building Block)



- Similar to D Flip Flop except
  - N-bit input and output
  - Write Enable input
- Write Enable:
  - Negated (or deasserted) (0): Data Out will not change
  - Asserted (1): Data Out will become Data In on rising edge of clock

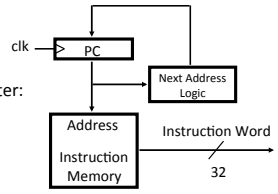
Storage Element: Register File



- Register File consists of 32 registers:
  - Two 32-bit output busses: busA and busB
  - One 32-bit input bus: busW
- Register is selected by:
  - RA (number) selects the register to put on busA (data)
  - RB (number) selects the register to put on busB (data)
  - RW (number) selects the register to be written via busW (data) when Write Enable is 1
- Clock input (clk)
  - Clk input is a factor ONLY during write operation
  - During read operation, behaves as a combinational logic block:
    - RA or RB valid ⇒ busA or busB valid after "access time."

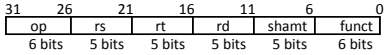
Step 3: Assemble DataPath Meeting Requirements

- Register Transfer Requirements ⇒ Datapath Assembly
- Instruction Fetch
- Read Operands and Execute Operation
- Common RTL operations
  - Fetch the Instruction: mem[PC]
  - Update the program counter:
    - Sequential Code: PC ← PC + 4
    - Branch and Jump: PC ← "something else"

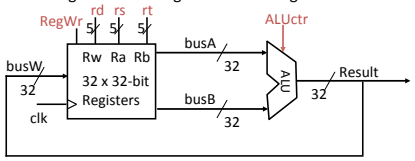


Step 3: Add & Subtract

- `R[rd] = R[rs] op R[rt]` (`addu rd,rs,rt`)
  - Ra, Rb, and Rw come from instruction's Rs, Rt, and Rd fields



  - ALUctr and RegWr: control logic after decoding the instruction



- … Already defined the register file & ALU

Agenda

- MIPS-lite Datapath
- Administrivia
- CPU Timing
- MIPS-lite Control

Who is in CS61c? Surprisingly Diverse Set of Majors

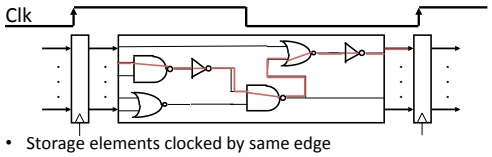
Applied Math	6	Film	1
Architecture	1	IEOR	2
Bioengineering	6	Japanese	2
Business Admin	9	Legal Studies	1
Chemical Eng	1	Linguistics	1
Chemistry	2	Mathematics	4
Civil Eng	1	Mechanical Eng	4
Cognitive Sci	11	OR & Mgt Sci	1
Computer Science	**3**	Physics	2
Double	***11***	Political Econ	1
Economics	6	Political Sci	2
EECS & Mat Sci	**2**	Statistics	3
Elec Engr Comp Sci	**227**	***Triple***	***1***
Elec Engr Physics	**1**	Undecl Eng	5
Engr-Mat Sci	1	**Undecl**	**149**

Fall 2012 enrollment > 2 x Fall 2011 enrollment!

Agenda

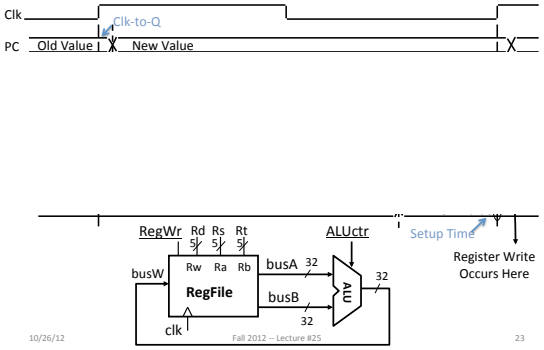
- MIPS-lite Datapath
- Administrivia
- CPU Timing
- MIPS-lite Control

Clocking Methodology

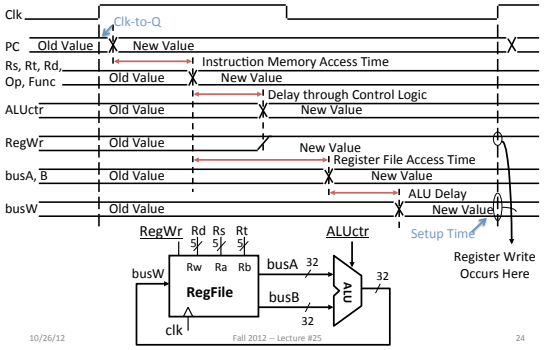


- Storage elements clocked by same edge
- "Critical path" (longest path through logic) determines length of clock period
- Have to allow for Clock-to-Q and Setup Times too
- This lecture (and P&H sections) 4.3-4.4 do whole instruction in 1 clock cycle for pedagogic reasons
  - Project 4 will do it in 2 clock cycles via simple pipelining
  - Soon explain pipelining and use 5 clock cycles per instruction

Register-Register Timing: One Complete Cycle



Register-Register Timing: One Complete Cycle

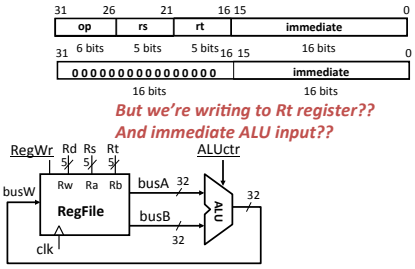


Agenda

- MIPS-lite Datapath
- Administrivia
- CPU Timing
- MIPS-lite Control

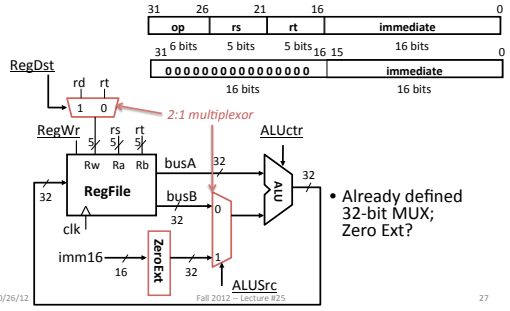
Logical Operations with Immediate

- R[rt] = R[rs] op ZeroExt[imm16]



Logical Operations with Immediate

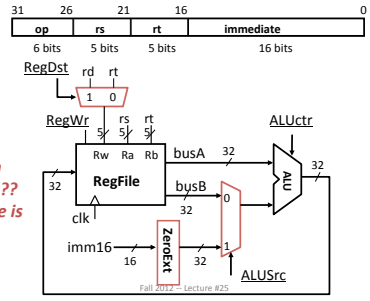
- R[rt] = R[rs] op ZeroExt[imm16]



- Already defined 32-bit MUX; Zero Ext?

Load Operations

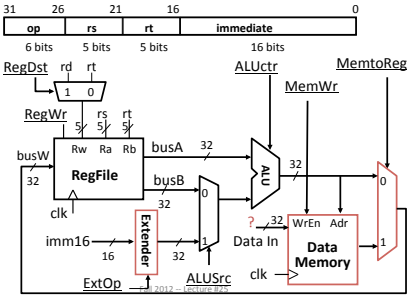
- R[rt] = Mem[R[rs] + SignExt[imm16]]
  Example: `lw rt,rs,imm16`



What sign extending?? And where is Mem??

Load Operations

- R[rt] = Mem[R[rs] + SignExt[imm16]]
  Example: `lw rt,rs,imm16`



And in Conclusion, … Single-Cycle Processor

- Five steps to design a processor:
  1. Analyze instruction set → datapath requirements
  2. Select set of datapath components & establish clock methodology
  3. Assemble datapath meeting the requirements
  4. Analyze implementation of each instruction to determine setting of control points that effects the register transfer.
  5. Assemble the control logic
     - Formulate Logic Equations
     - Design Circuits