

## 61C(?) In the News

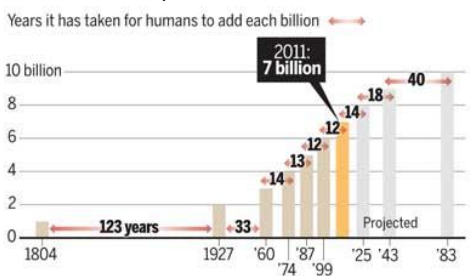
### World population hits 7 billion on Oct. 31, or thereabouts

Though it's impossible to say exactly when it will happen, demographers have chosen the date to mark the milestone. Humanity remains on a steep growth curve.

By Kenneth R. Weiss, Los Angeles Times  
October 31, 2011

Population grows when births exceed deaths. The 7-billion mark was reached because people are living longer and the number of infant deaths has dropped, because of a more secure food supply and because of advances in sanitation and medicine.

Up from 3 Billion in 1960



Population (Billion)	Year	Years to Add
1	1804	-
2	1927	123
3	'60	33
4	'74	14
5	'87	13
6	'99	12
7	2011	12
8	'25	14
9	'43	18
10	'83	40

Fall 2011 Sources: Earth Policy Institute, U.N. Department of Economic and Social Affairs  
MCCLATCHY-TRIBUNE

11/2/11

Fall 2011 Sources: Earth Policy Institute, U.N. Department of Economic and Social Affairs

MCCLATCHY-TRIBUNE

## CS 61C: Great Ideas in Computer Architecture (Machine Structures)

### Lecture 28: Single-Cycle CPU

#### *Datapath Control Part 1*

Instructors:  
Mike Franklin  
Dan Garcia

<http://inst.eecs.Berkeley.edu/~cs61c/fa11>

11/2/11

Fall 2011 -- Lecture #28

2

## Review

- CPU design involves Datapath, Control
  - 5 Stages for MIPS Instructions
    1. Instruction Fetch
    2. Instruction Decode & Register Read
    3. ALU (Execute)
    4. Memory
    5. Register Write
- Datapath timing: single long clock cycle or one short clock cycle per stage

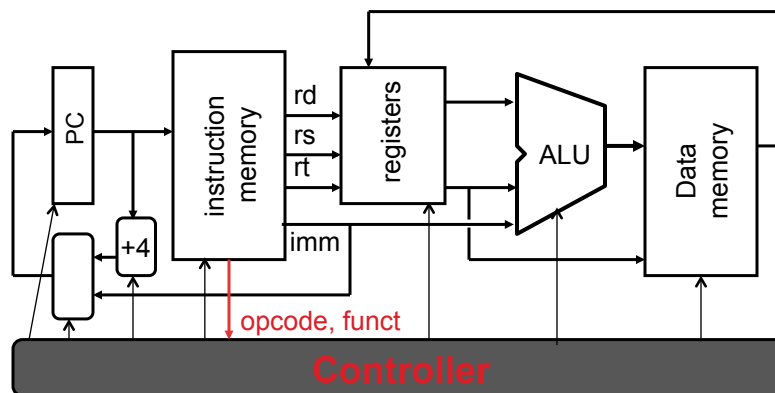
11/2/11

Fall 2011 -- Lecture #28

3

## Datapath and Control

- Datapath based on data transfers required to perform instructions
- Controller causes the right transfers to happen



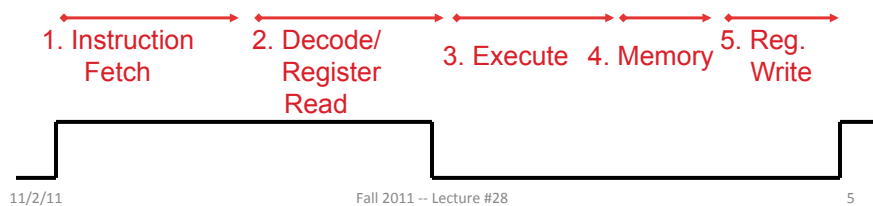
11/2/11

Fall 2011 -- Lecture #28

4

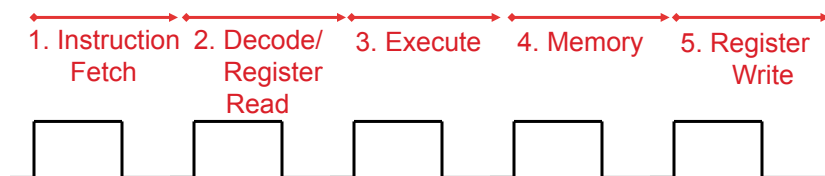
## CPU Clocking (1/2)

- For each instruction, how do we control the flow of information through the datapath?
- Single Cycle CPU: All stages of an instruction completed within one long clock cycle
  - Clock cycle sufficiently long to allow each instruction to complete all stages without interruption within one cycle



## CPU Clocking (2/2)

- Alternative multiple-cycle CPU: only one stage of instruction per clock cycle
  - Clock is made as long as the slowest stage



- Several significant advantages over single cycle execution:
  - Unused stages in a particular instruction can be skipped
  - OR instructions can be pipelined (overlapped)

## Agenda

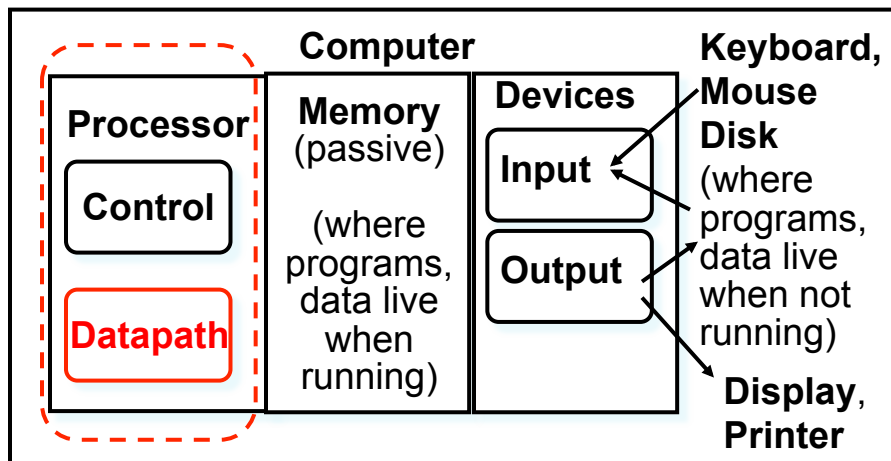
- Stages of the Datapath
- Datapath Instruction Walkthroughs
- Datapath Design

11/2/11

Fall 2011 -- Lecture #28

7

## Five Components of a Computer



11/2/11

Fall 2011 -- Lecture #28

8

## Processor Design: 5 steps

Step 1: Analyze instruction set to determine datapath requirements

- Meaning of each instruction is given by register transfers
- Datapath must include storage element for ISA registers
- Datapath must support each register transfer

Step 2: Select set of datapath components & establish clock methodology

Step 3: Assemble datapath components that meet the requirements

Step 4: Analyze implementation of each instruction to determine setting of control points that realizes the register transfer

Step 5: Assemble the control logic

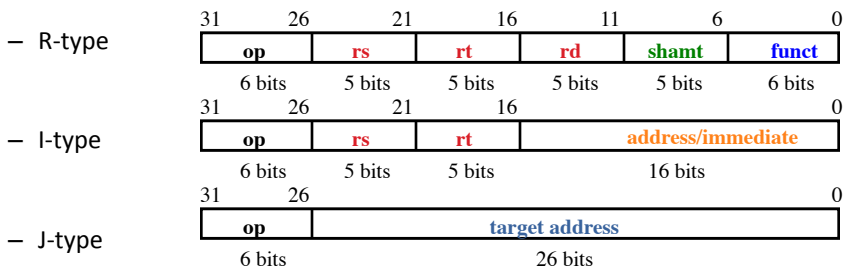
11/2/11

Fall 2011 -- Lecture #28

9

## The MIPS Instruction Formats

- All MIPS instructions are 32 bits long. 3 formats:



- The different fields are:
  - **op**: operation (“opcode”) of the instruction
  - **rs, rt, rd**: the source and destination register specifiers
  - **shamt**: shift amount
  - **funct**: selects the variant of the operation in the “op” field
  - **address / immediate**: address offset or immediate value
  - **target address**: target address of jump instruction

## The MIPS-lite Subset

- **ADDU and SUBU**

31	26	21	16	11	6	0				
<b>op</b>						<b>rs</b>	<b>rt</b>	<b>rd</b>	<b>shamt</b>	<b>funct</b>
6 bits		5 bits		5 bits		5 bits		5 bits		6 bits

- addu rd,rs,rt  
- subu rd,rs,rt
- **OR Immediate:**

31	26	21	16	0			
<b>op</b>				<b>rs</b>	<b>rt</b>	<b>immediate</b>	
6 bits		5 bits		5 bits		16 bits	

- ori rt,rs,imm16
- **LOAD and STORE Word**

31	26	21	16	0			
<b>op</b>				<b>rs</b>	<b>rt</b>	<b>immediate</b>	
6 bits		5 bits		5 bits		16 bits	

- lw rt,rs,imm16  
- sw rt,rs,imm16
- **BRANCH:**

31	26	21	16	0			
<b>op</b>				<b>rs</b>	<b>rt</b>	<b>immediate</b>	
6 bits		5 bits		5 bits		16 bits	

- beq rs,rt,imm16

11/2/11

Fall 2011 -- Lecture #28

11

## Register Transfer Language (RTL)

RTL gives the meaning of the instructions

All start by fetching the instruction

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

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

### 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
```

11/2/11

Fall 2011 -- Lecture #28

12

## Step 1: Requirements of the Instruction Set

- Memory (MEM)
  - Instructions & data (will use one for each)
- 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 registers?

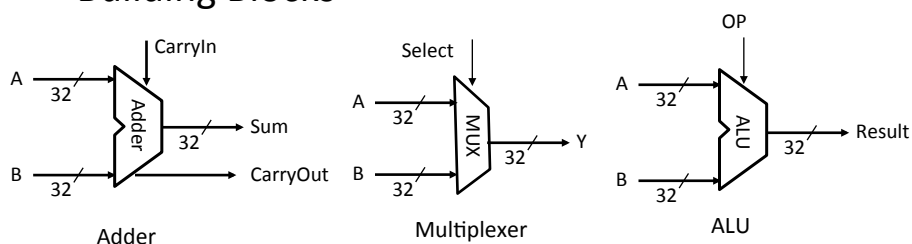
11/2/11

Fall 2011 -- Lecture #28

13

## Step 2: Components of the Datapath

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



11/2/11

Fall 2011 -- Lecture #28

14

## 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 follows Chapter 5

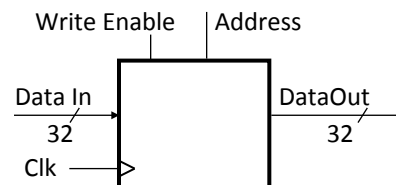
11/2/11

Fall 2011 -- Lecture #28

15

## 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  $\Rightarrow$  Data Out valid after “access time”



11/2/11

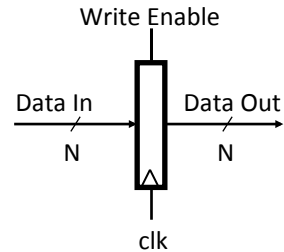
Fall 2011 -- Lecture #28

16



## 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 positive edge of clock



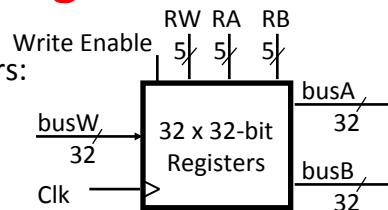
11/2/11

Fall 2011 -- Lecture #28

17

## 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  $\Rightarrow$  busA or busB valid after “access time.”



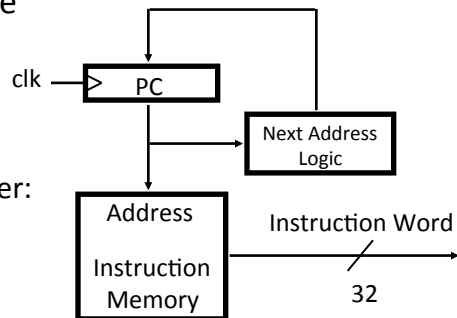
11/2/11

Fall 2011 -- Lecture #28

18

## Step 3a: Instruction Fetch Unit

- Register Transfer Requirements  $\Rightarrow$  Datapath Assembly
- Instruction Fetch
- Read Operands and Execute Operation
- Common RTL operations
  - Fetch the Instruction:  $mem[PC]$
  - Update the program counter:
    - Sequential Code:  $PC \leftarrow PC + 4$
    - Branch and Jump:  $PC \leftarrow$  “something else”



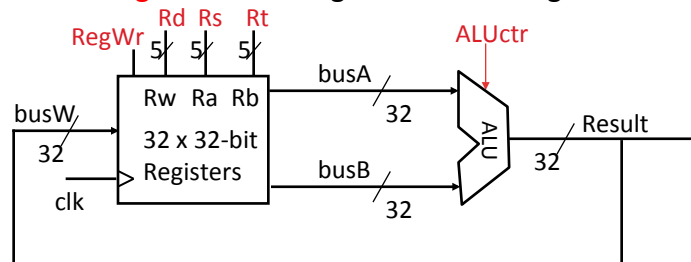
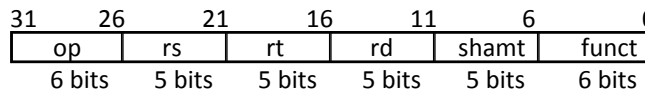
11/2/11

Fall 2011 -- Lecture #28

19

## Step 3b: Add & Subtract

- $R[rd] = R[rs] \text{ op } R[rt]$  (`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



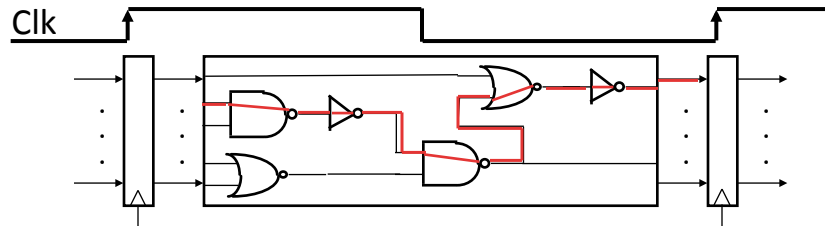
- ... Already defined the register file & ALU

11/2/11

Fall 2011 -- Lecture #28

20

## Clocking Methodology



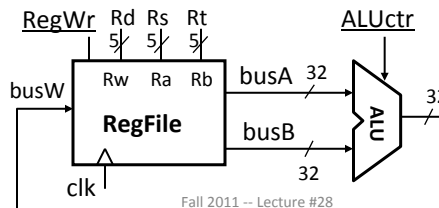
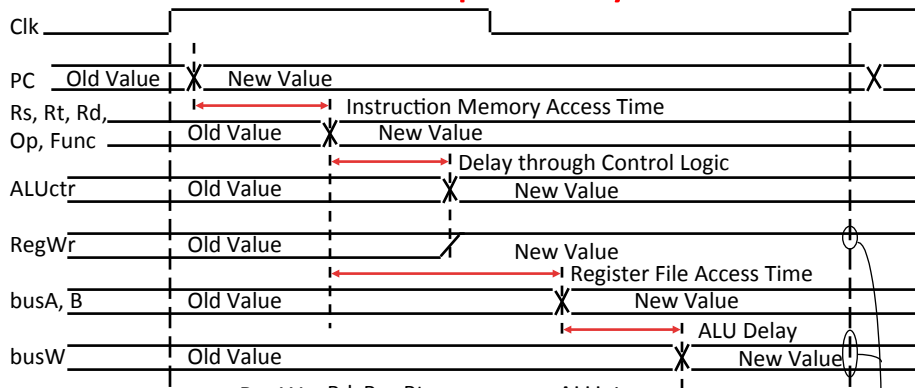
- Storage elements clocked by same edge
- Flip-flops (FFs) and combinational logic have some delays
  - Gates: delay from input change to output change
  - Signals at FF D input must be stable before active clock edge to allow signal to travel within the FF (set-up time), and we have the usual clock-to-Q delay
- “Critical path” (longest path through logic) determines length of clock period

11/2/11

Fall 2011 -- Lecture #28

21

## Register-Register Timing: One Complete Cycle

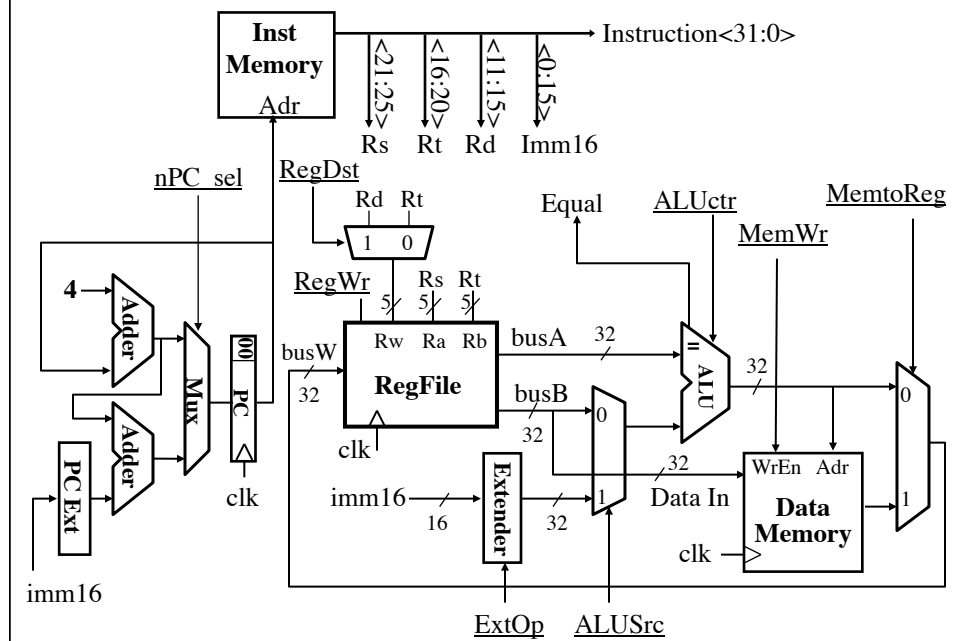


11/2/11

Fall 2011 -- Lecture #28

22

## Putting it All Together: A Single Cycle Datapath



## Processor Design: 3 of 5 steps

- Step 1: Analyze instruction set to determine datapath requirements
- Meaning of each instruction is given by register transfers
  - Datapath must include storage element for ISA registers
  - Datapath must support each register transfer
- Step 2: Select set of datapath components & establish clock methodology
- Step 3: Assemble datapath components that meet the requirements
- Step 4: Analyze implementation of each instruction to determine setting of control points that realizes the register transfer
- Step 5: Assemble the control logic