All Together: 
**Instruction Memory** + 
**Arithmetic Machine**

Pick up handout!!
Today’s lecture

- Instructions control the datapath
  - Instruction Memory
  - Program Counter (PC) is the address unit for instruction memory
  - Adder

- Putting all together
  - Arithmetic unit to work
What will $Q[3:0]$ be during the next clock cycle?

```
\begin{array}{cccc}
\text{D3} & \text{D2} & \text{D1} & \text{D0} \\
\text{Q3} & \text{Q2} & \text{Q1} & \text{Q0} \\
\end{array}
```

a) 0  b) 1  c) 3  d) 4  e) 5
Adder


D3  D2  D1  D0
Q3  Q2  Q1  Q0

reset  enable

Time

D[3:0]
Q[3:0]
reset
clk

0 1
0 1
1 1
0 1
0 1

a) 0x0
b) 0x2
c) 0x4
d) 0x6
e) 0x8
Previously...

- Register-to-register arithmetic instructions use the **R-type** format.

  \[
  \text{add } $5, $10, $4
  \]

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>shamt</th>
<th>func</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>6 bits</td>
</tr>
</tbody>
</table>

- Instructions with immediates all use the **I-type** format.

  \[
  \text{ori } $7, $2, 0x00ff
  \]

<table>
<thead>
<tr>
<th>op</th>
<th>rs</th>
<th>rt</th>
<th>immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>5 bits</td>
<td>5 bits</td>
<td>16 bits</td>
</tr>
</tbody>
</table>
Where are the instructions my program executes?

To look at the assembly code of a.out:

$ objdump –d a.out

The instructions executed by the program are in the .text section:

```
.text
main:
    addi $1, $0, 5
```
Programs require memory structures that are much larger than register files
Programs are stored in an instruction memory

We will read the memory but not modify it

```
.text
main:
  addi $1, $0, 5
  sub  $2, $1, $3
```
The instruction memory is byte addressable

- Addresses are 32-bits
  - # addresses: $2^{32} = 4$ Billion
- Each address contains 1 byte
  - Instructions are 4 bytes
    - occupy four contiguous locations
- Memory stores 4 GB (gigabytes)

$1 \text{ Byte} = 8 \text{ bits}$
MIPS instructions start at an address that is divisible by 4

- 0, 4, 8 and 12 are valid instruction addresses.
- 1, 2, 3, 5, 6, 7, 9, 10 and 11 are not valid instruction addresses.

\[
\text{word} = 9B \quad (32\text{b machine})
\]
A special register called Program Counter (PC) contains the address of the next instruction to execute.
Use an adder to increment PC to the next instruction

nextaddr = addr + 4
= PC + 4
Redrawn to match the MIPS diagram
Why aren’t 2 LSbs provided? 

a) Bug in the slide  
b) Memory is only $2^{30}$ big  
c) Bits [1:0] are always 2'b00 

d) Velociraptors ate them

Memory 

$2^{32} \times 8b$
MIPS datapath with a controlling instruction memory and program counter
Example

My program

$3 = 10$
$5 = -7$
$7 = $3 + $5$

What value will be stored in register 7 at the end of the program?

Assembly

```
addi $3, $0, 10
addi $5, $0, -7
add $7, $3, $5
```

a) -7  
b) 3  
c) 5  
d) 8  
e) 10
Example

My program

$3 = 10$
$5 = -7$
$7 = $3 + $5$

Assembly

Answer A
addi $3$, $0$, $0x000A$
subi $5$, $0$, $0x0007$
add $7$, $3$, $5$

Answer B
addi $3$, $0$, $0x000A$
addi $5$, $0$, $0xFFF9$
add $7$, $3$, $5$

Answer C
addi $3$, $0$, $0x000A$
addi $5$, $0$, $0xFFF8$
add $7$, $3$, $5$

Answer D
addi $3$, $0$, $0x000A$
sub $5$, $0$, $0x0007$
add $7$, $3$, $5$
Example

My program

$3 = 10$
$5 = -7$
$7 = 3 + 5$

Assembly

addi $3, 0, 0x000A$
addi $5, 0, 0xFFF9$
add $7, 3, 5$

Machine code

a) 00000
b) 00011
c) 00101
d) 00111
Little Endian - Least significant bits (little end) go first

Address
Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>0A</td>
</tr>
<tr>
<td>0x00000001</td>
<td>00</td>
</tr>
<tr>
<td>0x00000002</td>
<td>03</td>
</tr>
<tr>
<td>0x00000003</td>
<td>20</td>
</tr>
<tr>
<td>0x00000004</td>
<td>FF</td>
</tr>
<tr>
<td>0x00000005</td>
<td>FF</td>
</tr>
<tr>
<td>0x00000006</td>
<td>05</td>
</tr>
<tr>
<td>0x00000007</td>
<td>20</td>
</tr>
<tr>
<td>0x00000008</td>
<td>20</td>
</tr>
<tr>
<td>0x00000009</td>
<td>38</td>
</tr>
<tr>
<td>0x0000000A</td>
<td>25</td>
</tr>
<tr>
<td>0x0000000B</td>
<td>00</td>
</tr>
<tr>
<td>0x0000000C</td>
<td></td>
</tr>
<tr>
<td>0x0000000D</td>
<td></td>
</tr>
</tbody>
</table>

Assembly:

- addi $3, $0, 0x000A
- addi $5, $0, 0xFFF9
- add $7, $3, $5
- 0x00C53820
Big Endian – Most significant bits (big end) go first

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td></td>
</tr>
<tr>
<td>0x00000001</td>
<td></td>
</tr>
<tr>
<td>0x00000002</td>
<td></td>
</tr>
<tr>
<td>0x00000003</td>
<td></td>
</tr>
<tr>
<td>0x00000004</td>
<td>20  0x20  0xF9</td>
</tr>
<tr>
<td>0x00000005</td>
<td>05  0x05  0xFF</td>
</tr>
<tr>
<td>0x00000006</td>
<td>FF  0xFF  0x05</td>
</tr>
<tr>
<td>0x00000007</td>
<td>F9  0xF9  0x20</td>
</tr>
<tr>
<td>0x00000008</td>
<td></td>
</tr>
<tr>
<td>0x00000009</td>
<td></td>
</tr>
<tr>
<td>0x0000000A</td>
<td></td>
</tr>
<tr>
<td>0x0000000B</td>
<td></td>
</tr>
<tr>
<td>0x0000000C</td>
<td></td>
</tr>
<tr>
<td>0x0000000D</td>
<td></td>
</tr>
</tbody>
</table>

Assembly:
addi $3, $0, 0x000A
Machine: 0x2003000A

Assembly:  
addi $5, $0, 0xFFF9
Machine: 0x2005FFFF

Assembly:  
add $7, $3, $5
Machine: 0x00C53820

int *p;
char c = *(*(char*)p);
addi $3, $0, 0x0000A

MIPS decoder
- opcode[5:0]
- write_enable
- rd_src
- alu_src2
- except

ALU
- alu_op[2:0]
- rdWriteEnable
- rdData
- rdNum
- rtData
- rtNum
- rsData
- wr_enable
- rd_enable

Register File
- Rs
- Rd
- Rdest
- imm32
- imm16

Sign Extender
- in[15:0]
- out[31:0]
- imm32

Zero
- overflow
- zero
- negative

Zero
- overflow
- zero
- negative

Reg File
- 0
- 00000000
- 1
- ????
- 2
- ????
- 3
- ????
- 4
- ????
- 5
- ????
- 6
- ????
- 7
- ????
What decimal value is on the bus?  

a) -7  
b) 3  
c) 5  
d) 7  
e) 10