CS 3410
Computer System Organization and Programming

K. Walsh
kwalsh@cs

TAs:
Deniz Altinbuken
Hussam Abu-Libdeh

Consultants:
Adam Sorrin
Arseney Romanenko

If you want to make an apple pie from scratch, you must first create the universe.

— Carl Sagan
C

```c
int x = 10;
x = 2 * x + 15;
```

MIPS

```
addi r5, r0, 10
mul r5, r5, 2
addi r5, r5, 15
```

Machine code:

```
00100000000000001010000000000000001010
00000000000000101001010001000100000000
001000001010010100000000000001111
```
C

```c
int sum3(int v[]) {
    return v[0] +
    v[1] +
    v[2];
}

main() {
    ...
    int v[] = ...;
    int a = sum3(v);
    v[3] = a;
    ...
}
```
MIPS assembly language

sum3:
    lw    r9, 0(r5)
    lw    r10, 4(r5)
    lw    r11, 8(r5)
    add   r3, r9, r10
    add   r3, r3, r11
    jr     r31

main:
    ...
    addi  r5, r0, 1000
    jal    sum3
    sw     r3, 12(r5)
    ...

MIPS machine language

10001100101010010000000000000000000
10001100101010101000000000000001000
1000110010101010110000000000001000
000000010010101010001100000100000
000000001101011001100001000000
0000001111100000000000000001000
...
...
...
00100000000001010000001111101000
000011000001000000000000000000000
10101100101000110000000000001100
...

Computer System = ?
Input +
Output +
Memory +
Datapath +
Control

Registers
CPU

Video
Network
USB
Keyboard
Mouse

bus

Memory
Disk
Audio

Serial
FCED

r0: 0
r5:

1. Fetch
2. Decode
3. Execute

Control Unit

Function Unit

addi r5, r0, 10
muli r5, r5, 2
addi r5, r5, 15

CPU
addi r5, r0, 10
mul r5, r5, 2
addi r5, r5, 15
Fetch, Execute, Decode

1. Fetch
2. Decode
3. Execute

.raddi r5, r0, 10
.muli r5, r5, 2
.addi r5, r5, 15
addi r5, r0, 10
mul r5, r5, 2
addi r5, r5, 15
lw r9, 0(r5)
lw r10, 4(r5)
add r3, r9, r10
sw r3, 12(r5)
lw r9, 0(r5)
lw r10, 4(r5)
add r3, r9, r10
sw r3, 12(r5)
lw r9, 0(r5)
lw r10, 4(r5)
add r3, r9, r10
sw r3, 12(r5)
`lw r9, 0(r5)`  
`lw r10, 4(r5)`  
`add r3, r9, r10`  
`sw r3, 12(r5)`
Machine language represents program as numbers

• Store in / fetch from memory like other data
• 2 new registers:
  • Program counter (PC): address of next instruction
  • Instruction register (IR): current instruction

Revolutionary idea: a program is *just data*
  → von Neumann Architecture

Alternative:
• Separate memory systems for code and data
  → Harvard Architecture
1. Fetch @ PC
2. Update PC
3. Decode IR
4. Execute
### Instructions

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>lw r9, 4(r5)</td>
<td>0</td>
</tr>
<tr>
<td>addi r3, r9, 5</td>
<td>4</td>
</tr>
<tr>
<td>jr r31</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td>addi r5, r0, 1000</td>
<td>20</td>
</tr>
<tr>
<td>jal 0</td>
<td>24</td>
</tr>
<tr>
<td>sw r3, 12(r5)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>1004</td>
</tr>
<tr>
<td></td>
<td>1008</td>
</tr>
<tr>
<td></td>
<td>1012</td>
</tr>
</tbody>
</table>

### Control Unit

1. Fetch @ PC
2. Update PC
3. Decode IR
4. Execute

### Memory

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

### CPU

- r3:
- r5: lw
- r9:
- r10:
- r31: 28

### Function Unit

- IR: jal 0
- PC: 24 28 0
- MDR:
- MAR:

### Bus

- Connections between CPU and Memory
1. Fetch @ PC
2. Update PC
3. Decode IR
4. Execute

Control Unit

CPU
Function Unit

IR: jal
PC: 0x814

MDR:
MAR:

memory

bus

0: lw r9, 4(r5)
4: addi r3, r9, 5
8: jr r31
12: ...
16: ...
20: addi r5, r0, 1000
24: jal 0
28: sw r3, 12(r5)
32: ...
......
1000: 10
1004: 20
1008: 30
1012: 40
......
MIPS R3000 ISA (Instruction Set Architecture)
Interface between hardware and software
- memory: load, store, ...
- computational: add, sub, mul, ...
- control: jump, branch, ...
- floating point, cpu and memory management, ...

Instruction Formats

<table>
<thead>
<tr>
<th>OP</th>
<th>rs</th>
<th>rt</th>
<th>rd</th>
<th>sa</th>
<th>funct</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP</td>
<td>rs</td>
<td>rd</td>
<td>immediate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OP</td>
<td>address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Registers

RO - R31
PC
HI
LO
1-bit Multiplexor

\[
P \quad \text{MUX} \quad Q \quad S \quad R
\]

\[
\begin{array}{c}
P \\
Q \\
S \\
\end{array} \quad \begin{array}{c}
T \\
F \\
F \\
\end{array} \quad \begin{array}{c}
F \\
F \\
T \\
\end{array} \quad \begin{array}{c}
F \\
F \\
T \\
\end{array} 
\]

\[
\begin{array}{c}
\text{AND} \\
\text{OR} \\
\text{P} \\
\text{Q} \\
\text{R} \\
\end{array} 
\]

If $S$ is $T$
then $R = Q$

If $S$ is $F$
then $R = 0$
Computation
E.g. Multiplexor

State
E.g. Register

Gates
E.g. AND

Transistors
Why?
void A() {
    for (int i = 0; i < 4096; i++)
        for (int j = 0; j < 4096; j++)
            v[i][j] = f(v[i][j], i, j);
}

0.45 sec, (0.12 sec optimized)

void B() {
    for (int j = 0; j < 4096; j++)
        for (int i = 0; i < 4096; i++)
            v[i][j] = f(v[i][j], i, j);
    4.05 sec (3.52 sec optimized)
The number of transistors integrated on a single die will double every 24 months...

– Gordon Moore, Intel co-founder, 1965

1971 – 2300 transistors – 1MHz – 4004
1990 – 1M transistors – 50MHz – i486
2001 – 42M transistors – 2GHz – Xeon
2004 – 55M transistors – 3GHz – P4
2007 – 290M transistors – 3GHz – Core 2 Duo
2009 – 731M transistors – 2GHz – Nehalem
Xilinx FPGA

NVidia GPU

Berkeley mote
Why?

- Basic knowledge needed for *all* other areas of CS: operating systems, compilers, ...
- Levels are not independent
  - hardware design $\leftrightarrow$ software design $\leftrightarrow$ performance
- Crossing boundaries is hard but important
device drivers
- Good design techniques
  - abstraction, layering, pipelining, parallel vs. serial, ...
- Understand where the world is going
http://www.cs.cornell.edu/courses/cs3410

- Office Hours / Consulting Hours
- Lecture slides & schedule
- Logisim
- CSUG lab access (esp. second half of course)

Sections (choose one):

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>2:55 – 4:10pm</td>
<td>Hollister 110</td>
</tr>
<tr>
<td>W</td>
<td>3:35 – 4:50pm</td>
<td>Hollister 320</td>
</tr>
<tr>
<td>R</td>
<td>11:40 – 12:55pm</td>
<td>Hollister 401</td>
</tr>
<tr>
<td>R</td>
<td>2:55 – 4:10pm</td>
<td>Hollister 401</td>
</tr>
<tr>
<td>F</td>
<td>2:55 – 4:10pm</td>
<td>Snee 1150</td>
</tr>
</tbody>
</table>

- Will cover new material
- This week: intro to logisim
A) Love it
B) Okay
C) What?
D) Whatever
E) Please don’t
Grading:

• 4 Programming Assignments (35 – 45%)
  – Work in groups of two

• 2 Prelims (30 – 40%)

• 4-5 Homework Assignments (20 – 25%)
  – Work alone

• Discretionary (5%)
Academic Integrity:

• All submitted work must be your own (or your groups)
  – OK to study together, but do not share solutions
• Cite your sources

Stressed? Tempted? Lost?

• Come see me before due date!

Plagiarism in any form will not be tolerated