Microprocessors
Questions?
• Project 6: still need demos
• Project 7: due Thursday
Quiz
Components of a Microprocessor

• Memory:
  – Storage of data
  – Storage of a program
  – Either can be temporary or “permanent” storage

• Registers: small, fast memories
  – General purpose: store arbitrary data
  – Special purpose: used to control the processor
Components of a Microprocessor

- Instruction decoder:
  - Translates current program instruction into a set of control signals

- Arithmetic logical unit:
  - Performs both arithmetic and logical operations on data: add, subtract, multiply, AND, OR …

- Input/output control modules
Components of a Microprocessor

- Many of these components must exchange data with one another.
- It is common to use a ‘bus’ for this exchange.
Collections of Bits

- 8 bits: a “byte”
- 4 bits: a “nybble”
- “words”: can be 8, 16, or 32 bits (depending on the processor)
Collections of Bits

- A data bus typically captures a set of bits simultaneously
- Need one wire for each of these bits
- In the Atmel Mega2560: the data bus is 8-bits “wide”
- In your laptops: 32 or 64 bits
Memory

What are the essential components of a memory?
A Memory Abstraction

- We think of memory as an array of elements – each with its own address
- Each element contains a value
  - It is most common for the values to be 8-bits wide (so a byte)

```
<table>
<thead>
<tr>
<th>0x32</th>
<th>0xF1</th>
<th>0x11</th>
<th>0x67</th>
<th>......</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0x7B</td>
</tr>
</tbody>
</table>
```

\[2^{M-1}\]
A Memory Abstraction

- We think of memory as an array of elements – each with its own address
- Each element contains a value
  - It is most common for the values to be 8-bits wide (so a byte)

<table>
<thead>
<tr>
<th>Address</th>
<th>Stored value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x32</td>
<td>0xF1</td>
</tr>
<tr>
<td>0x11</td>
<td>0x67</td>
</tr>
<tr>
<td>0x7B</td>
<td></td>
</tr>
</tbody>
</table>

\[2^{M_{-1}}\]
Memory Operations

Read

```c
foo(A+5);
```

reads the value from the memory location referenced by the variable ‘A’ and adds the value to 5. The result is passed to a function called `foo();`
Memory Operations

Write

\[ A = 5; \]

writes the value 5 into the memory location referenced by ‘A’
Types of Memory

Many types of memory exist, including:

• Random access memory (RAM)
• Read-only memory (ROM)
  – Erasable/Programmable ROM (EPROM)
  – Electrically Erasable/Programmable ROM (EEPROM) (of FLASH memory)
Buses

• In the simplest form, a bus is a single wire
• Many different components can be attached to the bus
• Any component can take input from the bus or place information on the bus
Buses

• At most one component may write to the bus at any one time

• In a microprocessor, which component is allowed to write is usually determined by the code that is currently executing
Connecting Assembly Language to C

• Our C compiler is responsible for translating our code into Assembly Language
• Today, we rarely program in Assembly Language
  – Embedded systems are a common exception
  – Also: it is useful in some cases to view the assembly code generated by the compiler
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

The Assembly:

```assembly
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

………
**An Example**

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

Load the contents of memory location A into register 1

The Assembly:

```assembly
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

```
......
```
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

Load the contents of memory location B into register 2

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

……..

Andrew H. Fagg: Embedded Real-Time Systems: Microprocessors
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

Compare the contents of register 2 with those of register 1

This results in a change to the status register
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

Branch If Greater Than or Equal To:
jump ahead 3 instructions if true
An Example

A C code snippet:

```c
if (B < A) {
    D += A;
}
```

Branch if greater than or equal to will jump ahead 3 instructions if true

The Assembly:

1. `LDS R1 (A)`
2. `LDS R2 (B)`
3. `CP R2, R1`  
4. `BRGE 3`  
5. `LDS R3 (D)`  
6. `ADD R3, R1`  
7. `STS (D), R3`  

PC
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

Not true: execute the next instruction

The Assembly:

```
LDS R1 (A)  
LDS R2 (B)  
CP R2, R1   
BRGE 3      
LDS R3 (D)  
ADD R3, R1  
STS (D), R3 
```

......
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

Load the contents of memory location D into register 3

The Assembly:

```
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

PC
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

Add the values in registers 1 and 3 and store the result in register 3.

The Assembly:

```assembly
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
........
PC
```
An Example

A C code snippet:

```c
if(B < A) {
    D += A;
}
```

The Assembly:

```asm
LDS R1 (A)
LDS R2 (B)
CP R2, R1
BRGE 3
LDS R3 (D)
ADD R3, R1
STS (D), R3
```

Store the value in register 3 back to memory location D
Take-Aways

Instructions are the “atomic” actions that are taken by the processor

• Many different component work together to execute a single instruction
• One line of C code typically translates into a sequence of several instructions
• In the mega 2560, most instructions are executed in a single clock cycle

The high-level view is important here: you won’t be compiling programs on exams