Writing Assembly Code

- In Lab 7, you will write MIPS Assembly code
- You will use the MARS simulator to run your code

References

- H&H Chapter 6
- Lectures 9 and 10
- MIPS Cheat Sheet
An Example of MIPS Assembly Code

- Add all the even numbers from 0 to 10
  - $0 + 2 + 4 + 6 + 8 + 10 = 30$

High-level code

```c
int sum = 0;
for(int i = 0; i <= 10; i += 2) {
    sum += i;
}
```

MIPS assembly

```
# i=$s0; sum=$s1
addi $s0, $0, 0
addi $s1, $0, 0
addi $t0, $0, 12
loop:  beq $s0, $t0, done
       add $s1, $s1, $s0
       addi $s0, $s0, 2
       j    loop
done:
```
Recall: Arrays: Code Example

- We first load the base address of the array into a register (e.g., $s0) using \texttt{lui} and \texttt{ori}

### High-level code

```c
int array[5];

array[0] = array[0] * 2;

```

### MIPS assembly

```assembly
# array base address = $s0
# Initialize $s0 to 0x12348000
lui $s0, 0x1234
ori $s0, $s0, 0x8000

lw $t1, 0($s0)
sll $t1, $t1, 1
sw $t1, 0($s0)
lw $t1, 4($s0)
sll $t1, $t1, 1
sw $t1, 4($s0)
```
Lab 7: Exercise 1

- Write MIPS assembly code to compute the sum $A + (A + 1) + \cdots (B - 1) + B$, given two inputs $A$ and $B$.

- Example
  \[ A = 5, B = 10 \implies S = 5 + 6 + 7 + 8 + 9 + 10 = 45 \]

- For this exercise, you can use a subset of MIPS instructions: \texttt{ADD}, \texttt{SUB}, \texttt{SLT}, \texttt{XOR}, \texttt{AND}, \texttt{OR} and \texttt{NOR}, which are the instructions supported by the ALU you designed in the previous labs.

- Additionally, you are allowed to use \texttt{J}, \texttt{ADDI} and \texttt{BEQ}
Lab 7: Exercise 2

- Write MIPS assembly code to compute the **Sum of Absolute Differences** of two images

![Image](image1.png)

\[ S(x, y) = |I_1(x, y) - I_2(x, y)| \]

**Hints**

- Recall the **function calls** and the use of the **stack** in Lecture 10
- Read how to implement **recursive function calls** in H&H 6.4
In this lab, you will do what a compiler does: transforming high level code to MIPS assembly.

- Exercise 1: Write simple code and get familiar with the MARS simulator.
- Exercise 2: Sum of Absolute Differences of two images.
- Find Exercise 3 in the lab report.
Report Deadline

23:59, 21 May 2021
Backup Slides
# MIPS R-Type Instructions

<table>
<thead>
<tr>
<th>Description</th>
<th>Operation</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add two registers and store the result in a register $d$.</td>
<td>$d = s + t$; advance_pc (4);</td>
<td><code>add $d, $s, $t</code></td>
</tr>
<tr>
<td>Subtract $t$ from $s$ and store the result in $d$.</td>
<td>$d = s - t$; advance_pc (4);</td>
<td><code>sub $d, $s, $t</code></td>
</tr>
<tr>
<td>If $s$ is less than $t$, $d$ is set to one. $d$ gets zero otherwise.</td>
<td>if $s &lt; t$: $d = 1$; else: $d = 0$; advance_pc (4);</td>
<td><code>slt $d, $s, $t</code></td>
</tr>
<tr>
<td>Bitwise and of $s$ and $t$ and store the result in the register $d$.</td>
<td>$d = s &amp; t$; advance_pc (4);</td>
<td><code>and $d, $s, $t</code></td>
</tr>
<tr>
<td>Exclusive or of $s$ and $t$ and store the result in $d$.</td>
<td>$d = s ^ t$; advance_pc (4);</td>
<td><code>xor $d, $s, $t</code></td>
</tr>
<tr>
<td>Bitwise logic or of $s$ and $t$ and store the result in $d$.</td>
<td>$d = s | t$; advance_pc (4);</td>
<td><code>or $d, $s, $t</code></td>
</tr>
</tbody>
</table>
# MIPS I-Type Instructions

<table>
<thead>
<tr>
<th>Description:</th>
<th>Add sign-extended immediate to register $s and store the result in $t.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics:</td>
<td>$t = s + imm; PC=PC+4;</td>
</tr>
<tr>
<td>Syntax:</td>
<td>addi $t, $s, imm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description:</th>
<th>Branch if the contents of $s and $t are equal.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics:</td>
<td>if $s == $t: advance_pc (offset &lt;&lt; 2)); else: PC=PC+4;</td>
</tr>
<tr>
<td>Syntax:</td>
<td>beq $s, $t, offset</td>
</tr>
</tbody>
</table>
MIPS J-Type Instructions

<table>
<thead>
<tr>
<th>Description</th>
<th>Jump to the address.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantics</td>
<td>PC = nPC; nPC = (PC &amp; 0xf0000000)</td>
</tr>
<tr>
<td>Syntax</td>
<td>j target</td>
</tr>
</tbody>
</table>