# Intro to Web Assembly

## 240-316 3SA04

## วัตถุประสงค์

มีความเข้าใจในเรื่องของ "Web Assembly" (เว็บแอสเซมบลี) และวิธีการนำไปประยุกต์ใช้ในการพัฒนาแอปพลิเคชันบนเว็บเบราว์เซอร์อย่างมีประสิทธิภาพ

## บทนำ

เมื่อพัฒนาแอปพลิเคชันบนเว็บเบราว์เซอร์ นักพัฒนาโดยทั่วไปจะใช้ภาษาโปรแกรมที่ถูกแปลงเป็นภาษาเครื่อง (machine code) เช่น JavaScript ซึ่งเป็นภาษาสคริปต์ที่น่าเขียนและอ่านได้ง่าย อย่างไรก็ตาม ภาษา JavaScript มีประสิทธิภาพในการทำงานที่ต่ำกว่าภาษาเครื่อง ทำให้เกิดความล่าช้าในการทำงานบางอย่าง

Web Assembly (หรือเรียกสั้นๆ ว่า "Wasm") เป็นเทคโนโลยีที่ถูกพัฒนาขึ้นเพื่อแก้ปัญหาดังกล่าว โดย Wasm จะช่วยให้เราสามารถคอมไพล์ภาษาโปรแกรมอื่นๆ เป็นรหัส Wasm ซึ่งเป็นภาษาเครื่องที่มีประสิทธิภาพสูงกว่า JavaScript และสามารถทำงานได้เร็วกว่า นอกจากนี้ยังสามารถนำไปใช้ร่วมกับภาษาโปรแกรมอื่นๆ เช่น C, C++, และ Rust เพื่อสร้างแอปพลิเคชันที่สามารถทำงานได้ดีกว่าเดิมในเว็บเบราว์เซอร์

**ขั้นตอนทั่วไปในการประยุกต์ใช้ Wasm**

1. การทำงานของภาษาโปรแกรมระดับสูง: นักพัฒนาเขียนโค้ดในภาษาโปรแกรมระดับสูง เช่น C, C++, หรือ Rust ซึ่งเป็นภาษาที่มีประสิทธิภาพและมีประสิทธิภาพในการทำงาน
2. กระบวนการคอมไพล์: จากนั้นโค้ดต้นฉบับจะถูกคอมไพล์เป็นเว็บแอสเซมบลี (Web Assembly) โดยใช้คอมไพเลอร์ที่เฉพาะเจาะจง โดยที่คอมไพเลอร์นี้จะแปลงโค้ดระดับสูงเหล่านี้เป็นรหัสที่เป็นภาษาเครื่อง (bytecode) ที่เบราว์เซอร์สามารถเข้าใจได้
3. การโหลด Wasm ในเบราว์เซอร์: รหัสเว็บแอสเซมบลี (Wasm binary) ที่ถูกคอมไพล์แล้วจะถูกโหลดเข้าสู่เบราว์เซอร์เหมือนกับการโหลดไฟล์ JavaScript โดยพิเศษในหน้า HTML หรือที่เรียกว่า script tags หรืออาจดึงมาโดยใช้ JavaScript
4. การทำงานในเบราว์เซอร์: เมื่อรหัสเว็บแอสเซมบลี (Wasm binary) ได้ถูกโหลด มันจะถูกทำงานภายในเบราว์เซอร์ โดยสิทธิ์ของ Wasm จะถูกจำกัดให้ทำงานอยู่ในพื้นที่รักษาความปลอดภัย (sandboxed environment) เพื่อป้องกันไม่ให้มีผลกระทบต่อเว็บเพจหรือแอปพลิเคชันอื่นๆ

5. ระบบ Runtime และการจัดการหน่วยความจำ: เว็บแอสเซมบลี (Wasm) มีเครื่อง "เครื่องเสมือน" ที่ใช้ในรันรหัสเว็บแอสเซมบลีอย่างมีประสิทธิภาพ และมีระบบการจัดการหน่วยความของตนเองโดยไม่ต้องมีการเข้าถึงหน่วยความจำของเครื่องโดยตรง
6. การเชื่อมโยงกับ JavaScript: เว็บแอสเซมบลี (Wasm) สามารถทำงานร่วมกับภาษา JavaScript ได้อย่างราบรื่น นักพัฒนาสามารถเรียกใช้ฟังก์ชัน Wasm จาก JavaScript และกลับกัน

**ประโยชน์ของการใช้ Wasm**

1. ประสิทธิภาพ: หนึ่งในความสำคัญของ Wasm คือความเร็วของการทำงาน โดยเนื่องจากเป็นรหัสที่เป็นภาษาเครื่อง (binary format) ทำให้ Wasm สามารถทำงานได้เร็วกว่ารหัส JavaScript แบบดั้งเดิม สิ่งนี้เป็นที่ต้องการสำหรับงานที่มีการคำนวณที่ซับซ้อนและเรื่องที่ต้องการประสิทธิภาพสูง อาทิเช่น เกม โปรแกรมจำลอง และการประมวลผลสื่อ
2. ความปลอดภัย: เว็บแอสเซมบลี (Wasm) ทำงานภายในแซนด์บอกซ์ที่มีความปลอดภัย ซึ่งช่วยป้องกันไม่ให้เกิดการเข้าถึงข้อมูลของผู้ใช้งานที่เป็นสิ่งสำคัญหรือทำลายระบบของผู้ใช้งาน รูปแบบความปลอดภัยนี้ช่วยให้ Wasm เป็นที่ปลอดภัยในการทำงานบนเว็บเพจ
3. ความเป็นอิสระในการใช้งานบนแพลตฟอร์ม: เว็บแอสเซมบลี (Wasm) ถูกออกแบบมาเพื่อให้เป็นอิสระในการใช้งานบนแพลตฟอร์มที่แตกต่างกัน นั่นคือรหัสเว็บแอสเซมบลี (Wasm binary) เดียวกันสามารถทำงานได้บนแพลตฟอร์มและโครงสร้างต่างๆ โดยไม่ต้องแก้ไข

# ตอนที่ 1 พาเดินชม Web Assembly

ติดตั้งเครื่องมือ LLVM ในการใช้คอมไพล์ C ไปเป็น Web Assembly

- Windows: choco install llvm
- OSX: brew install llvm
- Linux: ทำตามขั้นตอนในเว็บ https://apt.llvm.org/

**** ในกรณีที่มีปัญหาในการติดตั้ง LLVM นศ. สามารถใช้ไฟล์ wasm ที่แปลงเรียบร้อยได้จาก LMS**

1. เขียนโค้ดภาษา C ตามข้างล่าง โดยในตอนนี้จะเป็นการเขียนฟังก์ชันสำหรับหาผลลัพธ์ของการยกกำลัง ในภาษา C

```
int square(int n) {
   return n*n;
}
```

2. คอมไพล์ C เป็น wat (ไฟล์ Web Assembly ในรูปแบบข้อความ) และ assemble เป็น wasm (ไฟล์ Web Assembly ในรูปแบบไบนารี) ตามลำดับ

   ```
   > clang --target=wasm32 --no-standard-libraries -Wl,--export-all -Wl,--no-entry -o findsquare.wasm findsquare.c
   ```

3. ถัดไปให้สร้างไฟล์ โฟลเดอร์ขึ้นมาและนำไฟล์ที่ได้จากการดาวน์โหลดเมื่อสักครู่ไปวางเอาไว้ พร้อมสร้างไฟล์ main.html โดยให้เขียนโค้ดเพื่อดาว์นโหลดไฟล์ wasm ด้วยฟังก์ชัน fetch เพื่อนำมาใช้ในเพจ main.html

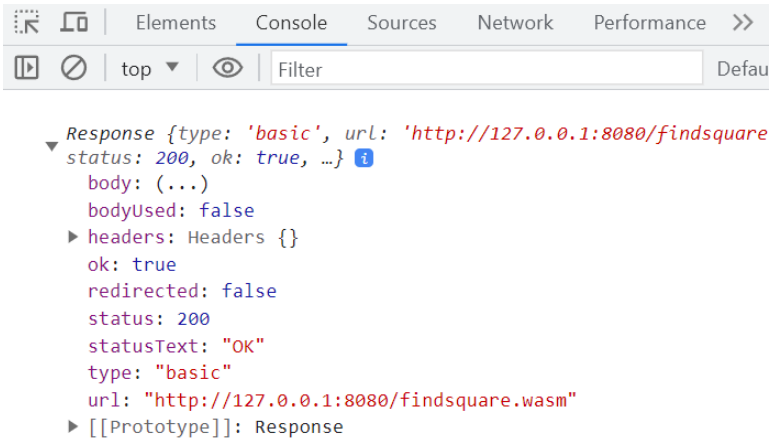
```
<script>
  async function loadWasm(){
    let result = await fetch("findsquare.wasm")
    console.log(result)
  }
  loadWasm()
</script>
```

4. ทำการสร้างเว็บเซอร์ฟเวอร์จำลอง เพื่อให้บริการไฟล์ html และ wasm ที่ได้สร้างขึ้น (การใช้คำสั่งด้านล่างจำเป็นต้องมี Node.js ติดตั้งอยู่ในเครื่อง) โดยให้สั่ง command ด้านล่าง ในขณะที่ Working Directory ปัจจุบันอยู่ที่เดียวกันกับโฟลเดอร์ที่เก็บไฟล์ html และ wasm

```
$ npx http-server
```

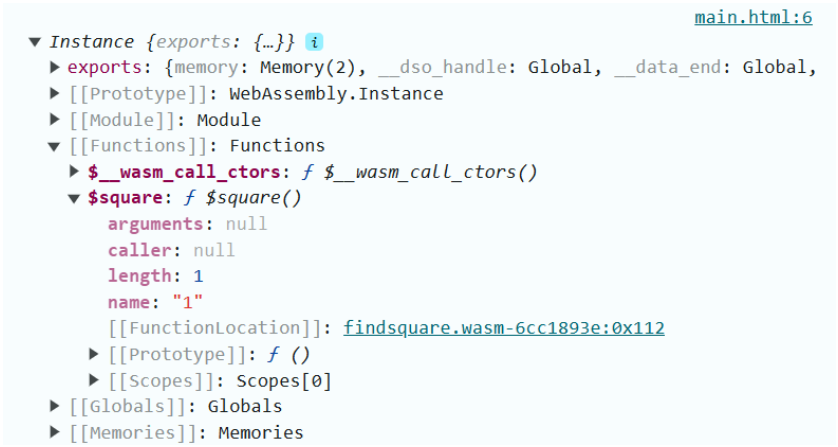
   ในกรณีคำสั่งข้างต้นรันได้สำเร็จ จะแสดง URL เพื่อเข้าถึงไฟล์ข้างต้น ตัวอย่างเช่น http://127.0.0.1:8080

5. ใช้ Chrome Browser (หรือเทียบเคียง) เข้าถึงไฟล์ main.html ที่ URL http://127.0.0.1:8080/main.html หลังจากนั้นให้ทำการเปิด Developer Tools แล้วไปที่แท็บ Console ที่ดูผลลัพธ์




6. ปรับฟังก์ชัน loadWasm() ให้ทำการ compile และสร้าง instance ของ wasm เพื่อเตรียมพร้อมสำหรับการ execute ฟังก์ชัน square ที่ได้เขียนขึ้น ทั้งนี้ให้สังเกตุผลลัพธ์จาก Developer Tools

```
<script>
  async function loadWasm(){
    let result = await fetch("findsquare.wasm")
    let compiledmod = await WebAssembly.compile(await result.arrayBuffer())
    let instance = new WebAssembly.Instance(compiledmod)
    console.log(instance)
  }
  loadWasm()
</script>
```




7. ทดลองเรียกใช้ฟังก์ชัน Square ภายใต้ชื่อฟังก์ชันที่ปรากฏในผลลัพธ์ข้างต้น (อาจจะใช้ console.log หรือเขียน UI เพื่อเรียกใช้งาน) เช่น

```
instance.exports.square(13)
```

# ตอนที่ 2 สนุกกับหน่วยความจำ (ลุยด้วยตัวเอง)

อ้างอิง: https://www.tutorialspoint.com/webassembly/webassembly_hello_world.htm

1. สร้าง Wasm จากโค้ดภาษา C ด้านล่าง

```
char *c_hello() {
   return "Hello World";
}
```

2. ทดสองเรียกใช้ฟังก์ชันข้างต้นจาก JavaScript โดยตรงสังเกตุผลลัพธ์ที่ฟังก์ชันคืนมา
3. ทดลองเข้าถึงข้อมูลผ่าน memory ของ wasm โดยใช้โค้ดข้างล่าง รันและสังเกตุผล

```
let ref = instance.exports.c_hello()
let mytext = "";
for (let i=ref; buffer[i]; i++){
  mytext += String.fromCharCode(buffer[i]);
}
console.log(mytext)
```

4. สังเกตุผล อภิปรายกับเพื่อนร่วมคลาส และหาข้อมูลเพิ่มเติมเพื่ออธิบายสิ่งที่เกิดขึ้น

# งานท้ายการทดลอง

ให้นักศึกษาเขียนรายงานผลการเรียนรู้ โดยมีเนื้อหา 3 ตอน นำส่งใน LMS ในรูปแบบของ PDF

- ตอนที่ 1: สรุปความรู้ที่ได้จากการทดลอง (ความยาวประมาณ 1 หน้ากระดาษ A4)
- ตอนที่ 2: ความคิดเห็นของตนเองต่อตัวเทคโนโลยี Wasm (ความยาว 1-2 ย่อหน้า)
- ตอนที่ 3: ตัวอย่างการนำ Wasm ที่มีการนำไปใช้งาน 1 ตัวอย่าง (ระบุแหล่งที่มาของข้อมูล)

*หมายเหตุ การให้คะแนน*

1. Checkpoint จาก TA จำนวน 20 คะแนน
2. รายงาน 30 คะแนน
3. คะแนนสอบ 50 คะแนน (เนื้อหาครอบคลุม lecture ก่อนลงแลบ และเนื้อหาในแลบนี้)