1.1) What does the code hint about the kind of instruction set? (e.g. Accumulator, Register Memory, Memory Memory, Register Register) Please justify your answer.

Ans ดูจากการใส่ค่าลงไปในตัวแปร ยกตัวอย่างเช่น “movl -4(%rbp), %eax” คือการเก็บค่าจาก memory -4(%rbp) เพื่อไปเก็บลงใน register eax ซึ่งเป็นการเก็บโดยใช้แบบ Memory Register แต่ก็มีการเก็บค่าแบบ Register Register ด้วยเช่น “movl %eax, %ecx” คือการเก็บค่าจาก register eax ไปยัง register ecx

1.2) Can you tell whether the architecture is either Restricted Alignment or Unrestricted Alignment? Please explain how you come up with your answer

Ans Restricted alignment จาก code

```
movl    %ecx, 16(%rbp)
movl    %edx, 24(%rbp)
```

พบว่าการจอง memory จะทำการจองทีละพหุคูณของ 8

1.3) What does the result suggest regarding the register saving (caller save vs. callee save)? Please provide your analysis.

Ans

```
testmax:
    pushq   %rbp
    .seh_pushreg    %rbp
    movq    %rsp, %rbp
    .seh_setframe   %rbp, 0
    subq    $32, %rsp
    .seh_stackalloc 32
    .seh_endprologue
    movl    %ecx, 16(%rbp)
    movl    %edx, 24(%rbp)
    movl    24(%rbp), %eax
    movl    %eax, %edx
    movl    16(%rbp), %ecx
    call    max1
    addq    $32, %rsp
    popq    %rbp
    ret
```

จาก code จะเห็นได้ว่าก่อนที่จะมีการเรียก max1 จะมีการ PASS โดยการ ใส่ไว้ใน register eax และ edx และ max1 จะทำการเรียกใช้ค่าที่ PASS โดยใช้ eax และ edx ซึ่งคือ Caller save

```
max1:
    pushq   %rbp
    .seh_pushreg    %rbp
    movq    %rsp, %rbp
    .seh_setframe   %rbp, 0
    .seh_endprologue
    movl    %ecx, 16(%rbp)
    movl    %edx, 24(%rbp)
    movl    24(%rbp), %edx
    movl    16(%rbp), %eax
    cmpl    %eax, %edx
    cmovge  %edx, %eax
    popq    %rbp
    ret
    .seh_endproc
    .globl  testmax
    .def    testmax;    .scl    2;  .type   32; .endef
    .seh_proc   testmax
```

แต่ว่าใน max1 จะทำการ เก็บ rbp (register base pointer) ไว้ เมื่อฟังก์ชันทำงานจบ จะทำการย้ายกลับไปทำงานต่อที่ testsmax ซึ่งเป็น Callee save

1.4) How do the arguments be passed and the return value returned from a function? Please explain the code.

```
testmax:
    pushq   %rbp
    .seh_pushreg    %rbp
    movq    %rsp, %rbp
    .seh_setframe   %rbp, 0
    subq    $32, %rsp
    .seh_stackalloc 32
    .seh_endprologue
    movl    %ecx, 16(%rbp)
    movl    %edx, 24(%rbp)
    movl    24(%rbp), %eax
    movl    %eax, %edx
    movl    16(%rbp), %ecx
    call    max1
    addq    $32, %rsp
    popq    %rbp
    ret
```

จาก code จะเห็นได้ว่าก่อนที่จะมีการเรียก max1 จะมีการ PASS โดยการ ใส่ไว้ใน register eax และ edx และ max1 จะทำการเรียกใช้ค่าที่ PASS โดยใช้ eax และ edx

```
max1:
    pushq   %rbp
    .seh_pushreg    %rbp
    movq    %rsp, %rbp
    .seh_setframe   %rbp, 0
    .seh_endprologue
    movl    %ecx, 16(%rbp)
    movl    %edx, 24(%rbp)
    movl    24(%rbp), %edx
    movl    16(%rbp), %eax
    cmpl    %eax, %edx
    cmovge  %edx, %eax
    popq    %rbp
    ret
    .seh_endproc
    .globl  testmax
    .def    testmax;    .scl    2;  .type   32; .endef
    .seh_proc   testmax
```

และการ return ค่า จะทำการส่งผ่าน register eax

1.5) Find the part of code (snippet) that does comparison and conditional branch. Explain how it works.

Ans

```
cmpl    24(%rbp), %eax
setg    %al
```

Code นี้เป็นส่วนของการเทียบระหว่างค่าที่อยู่ใน memory 24(%rbp) กับค่าที่อยู่บน Register eax หากค่าที่ 24(%rbp) มากกว่า eax ใส่ค่า 1 ไว้ใน register al แต่ถ้าไม่จะเป็น 0

```
movzbl  %al, %eax
movl    %eax, -8(%rbp)
cmpl    $0, -8(%rbp)
je  .L4
movl    16(%rbp), %eax
movl    %eax, -4(%rbp)
jmp .L5
```

หลังจากที่ได้ค่า al มาแล้วจะทำการเก็บไว้ใน eax และย้ายไป -8(%rbp) หลังจากนั้นจะนำค่านั้นมาเทียบกับ 0 หากมีค่าเท่ากันจะไปทำ L4 หากไม่จะทำการนำอีกค่าที่มากกว่ามาใส่ใน -4(%rbp) และไปทำ L5 (If else)

1.6) What are the differences that you may observe from the result (as compared to that without optimization). Please provide your analysis

Ans หลังจากที่มีการ Optimization จะเหลือเพียงไม่กี่บรรทัดและไม่ทำการเก็บค่าลงบน Memory คำนวนและเก็บลงบน Register อย่างเดียว นอกจากนี้ max1 และ max2 ยังเหมือนกัน

1.7) Please estimate the CPU Time required by the max1 function (using the equation CPU time=ICxCPIxTc). If possible, create a main function to call max1 and use the time command to measure the performance. Compare the measure to your estimation. What do you think are the factors that cause the difference? Please provide your analysis.

Ans จาก optimized max1 IC = 4 และ CPI ของ intel คือ 1 / 2.5 = 0.4 และ Tc คือ 1/4GHz = 2.5*10^(-10) จะได้ CPU time = 4 * 0.4 * 2.5*10^(-10) = 4*10^(-10) s

```
real    0m0.037s
user    0m0.000s
sys     0m0.000s
```

เป็น 0.00s เพราะเวลาน้อยมากเกินไป

2.)

No optimized = (8.531 + 8.437 + 8.609) /3 =8.525 s

O1 = (6.765 + 6.748 + 6.739)/3 = 6.750 s

O2 = (3.437 + 3.425 + 3.591)/3 = 3.484 s

O3 = (3.609 + 3.455 + 3.526)/3 = 3.530 s

3.) O1

- อาจมีการปรับปรุงพื้นฐานเช่นการจัดสรรทะเบียนพื้นฐานและการจัดเรียงคำสั่งใหม่

- การเรียกซ้ำอาจยังคงอยู่ แต่อาจมีการปรับปรุงโครงสร้างและประสิทธิภาพ

- ถึงแม้จะมีการปรับปรุงมากกว่าแบบไม่ถูกปรับให้เหมาะสม แต่ไม่ได้เป็นระดับการปรับปรุงที่ให้ความสำคัญกับประสิทธิภาพของโค้ด

O2 จะไม่มี Recursion แต่จะใช้ Memory ในการทำ DP

O3 การเรียกใช้รูปแบบแบบลูปมักถูกนำมาใช้แทนการเรียกใช้ลูปที่ลึก ลดการใช้สแต็กอย่างมากและเพิ่มประสิทธิภาพ