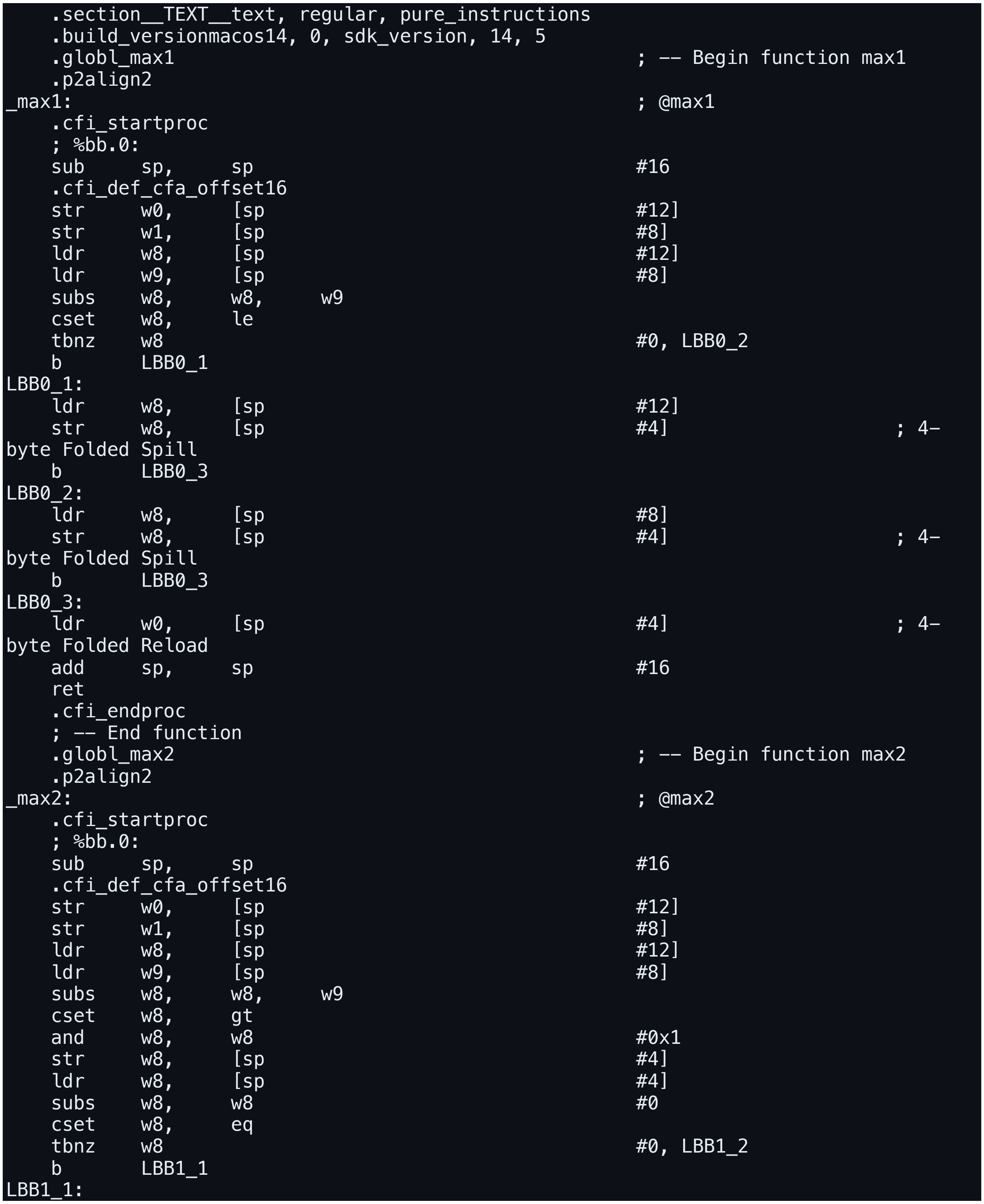
Assignment # 1 6532037221

Assembly code for max.c



```
    .section__TEXT__text, regular, pure_instructions
    .build_versionmacos14, 0, sdk_version, 14, 5
    .globl_max1                                         ; -- Begin function max1
    .p2align2
_max1:                                                  ; @max1
    .cfi_startproc
    ; %bb.0:
    sub     sp,     sp                                  #16
    .cfi_def_cfa_offset16
    str     w0,     [sp                                 #12]
    str     w1,     [sp                                 #8]
    ldr     w8,     [sp                                 #12]
    ldr     w9,     [sp                                 #8]
    subs    w8,     w8,     w9
    cset    w8,     le
    tbnz    w8                                          #0, LBB0_2
    b       LBB0_1
LBB0_1:
    ldr     w8,     [sp                                 #12]
    str     w8,     [sp                                 #4]                    ; 4-
byte Folded Spill
    b       LBB0_3
LBB0_2:
    ldr     w8,     [sp                                 #8]
    str     w8,     [sp                                 #4]                    ; 4-
byte Folded Spill
    b       LBB0_3
LBB0_3:
    ldr     w0,     [sp                                 #4]                    ; 4-
byte Folded Reload
    add     sp,     sp                                  #16
    ret
    .cfi_endproc
    ; -- End function
    .globl_max2                                         ; -- Begin function max2
    .p2align2
_max2:                                                  ; @max2
    .cfi_startproc
    ; %bb.0:
    sub     sp,     sp                                  #16
    .cfi_def_cfa_offset16
    str     w0,     [sp                                 #12]
    str     w1,     [sp                                 #8]
    ldr     w8,     [sp                                 #12]
    ldr     w9,     [sp                                 #8]
    subs    w8,     w8,     w9
    cset    w8,     gt
    and     w8,     w8                                  #0x1
    str     w8,     [sp                                 #4]
    ldr     w8,     [sp                                 #4]
    subs    w8,     w8                                  #0
    cset    w8,     eq
    tbnz    w8                                          #0, LBB1_2
    b       LBB1_1
LBB1_1:
    ldr     w8,     [sp                                 #12]
    str     w8,     [sp]
    b       LBB1_3
LBB1_2:
    ldr     w8,     [sp                                 #8]
    str     w8,     [sp]
    b       LBB1_3
LBB1_3:
    ldr     w0,     [sp]
    add     sp,     sp                                  #16
    ret
    .cfi_endproc
    ; —— End function
    .subsections_via_symbols
```

1)

```
    str     w1,     [sp                                 #8]
    ldr     w8,     [sp                                 #12]
```

- From this part of code, it’s considered as **Register-Memory** because it loads and stores exchange between memory (sp = stack pointer) and register (w1 and else for ARM)

- From the offset value of the stack pointer, we can verify that it’s **restricted alignment** since every address was offset by multiple of 32-bit integer (4 bytes) if its unrestricted alignment is not.

```
_testMax:                                               ; @testMax
    .cfi_startproc
    ; %bb.0:
    sub     sp,     sp                                  #32
    .cfi_def_cfa_offset32
    stp     x29,    x30,    [sp                         #16]            ; 16—byte
Folded Spill
    add     x29,    sp                                  #16
    .cfi_def_cfaw29, 16
    .cfi_offsetw30, —8
    .cfi_offsetw29, —16
    stur    w0,     [x29                                #—4]
    str     w1,     [sp                                 #8]
    ldur    w0,     [x29                                #—4]
    ldr     w1,     [sp                                 #8]
    bl      _max1
    ldp     x29,    x30,    [sp                         #16]            ; 16—byte
Folded Reload
    add     sp,     sp                                  #32
    ret
    .cfi_endproc
    ; —— End function
    .subsections_via_symbols
```

- They both have caller-saved and callee-saved.
  the caller (_testmax) save the register (x29 and x30) restore them later after the function called. [caller-saved]
  the caller declares the register (w0 and w1) to be stored and it's passed across the function via these register after callee finished their execution these still be the same [callee-saved]

- It uses w0 and w1 register to pass arguments to the max1 function after that the bl that link x30 to keep the return address from _max1 to the _testmax

Testmax code (partial) storing and loading to w0 and w1

```
    stur    w0,     [x29                                #-4]
    str     w1,     [sp                                 #8]
    ldur    w0,     [x29                                #-4]
    ldr     w1,     [sp                                 #8]
    bl      _max1
```

max1 (partial) storing from w0, w1 to sp+12 & sp+8.

```
    str     w0,     [sp                                 #12]
    str     w1,     [sp                                 #8]
```

- The snippet that making the comparison and conditional branch.

```
subs    w8,     w8,     w9
cset    w8,     le
tbnz    w8                                      #0, LBB0_2
b       LBB0_1
```

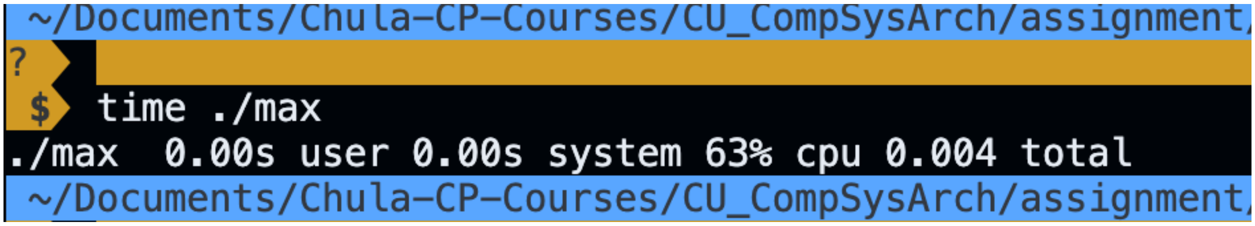
the code above the w8 got the result of subtraction between w8 and w9 (from max.c assembly ode at the first page) coming to condition to test whether it should jump to which block next e.g., LBB0_1 or LBB0_2.

After creating max.s file with gcc -O2 -S max.c, the optimization one (O2) will not keep the value after the comparison by using subtraction but use the cmp instruction to get only the flag to branch to the flag condition making then using the csel to get the comparison result from its flag.

Compiling with O2

IC = 3, CPI = 1, Tc = 0.3125 ns

Then the CPU time should be 0.9375 ns



```
 ~/Documents/Chula-CP-Courses/CU_CompSysArch/assignment
?
 $  time ./max
./max  0.00s user 0.00s system 63% cpu 0.004 total
 ~/Documents/Chula-CP-Courses/CU_CompSysArch/assignment
```

Since the hardware is not as fast as ideal, process management of the CPU in the device and other factor may cause it is slower than we expected.

2)

[Apple clang version 15.0.0 (clang-1500.3.9.4)]
From each optimization in fibo program, it can be demonstrated the time in table below.

optimization	user	sys	total
O0	10.90	0.05	11.149
	10.85	0.03	10.959
	10.86	0.04	10.915
O1	6.02	0.01	6.053
	6.03	0.02	6.105
	6.03	0.03	6.082
O2	6.20	0.01	6.222
	6.22	0.02	6.281
	6.22	0.04	6.296
O3	6.00	0.01	6.020
	6.01	0.04	6.068
	6.04	0.03	6.108

Avg of O0 = 11.00 s (total time), 10.87(user)

Avg of O1 = 6.08 s (total time), 6.03(user)

Avg of O2 = 6.27 s (total time), 6.21(user)

Avg of O3 = 6.07 s (total time), 6.02(user)

O0 > O2 > O1 > O3 as a result, the O3 is the best optimization, but surprisingly it has the execution time near the O1 optimization.

3)

- O0 is not optimization is the default code
- O1 Moderate some optimization but not change compilation time too much
- O2 performing all nearly support optimizations that do not involve space-speed trade-off, it increases compilation time and performance, loop-vectorize
- O3 aggressively optimization more than O2 with more options.