**Spring 2024**
**Exam II**
**2024-04-05**

---

# Instructions

This exam contains 8 pages (including this cover page) and 22 questions. It is out of 76 points.

You have **50 minutes** to complete the examination. As a courtesy to your classmates, we ask that you not leave during the last 15 minutes.

For this exam, you have been given a separate answer sheet to fill in your responses.

**DO** shade in the bubbles on your answer sheet without going outside the lines

**DO** feel free to write on this exam packet

**DO** assume all multiple-choice questions on this test are single-select **unless otherwise indicated**

**DO NOT** write anything on your answer sheet except for your name, computing ID, signature, and answers in the designated areas

**DO NOT** use a calculator, consult notes, or collaborate with classmates

We will use the following data type sizes:

| x86-64 Suffix | C Types | size in bits |
|---|---|---|
| `b` | `char` | 8 |
| `w` | `short` | 16 |
| `l` | `int` and `float` | 32 |
| `q` | `long` and `double` | 64 |

Function arguments are in (in order) `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8`, `%r9`; return values are in `%rax`.

The next page contains reference material which you are welcome to refer to during the test if you would like.

## Our Example ISA

*This is the same ISA used in HW03 and HW04, but presented to fit onto one printed page.*

Each instruction is one or two bytes, with the meaning of those bytes being:


Not all instructions have the second byte; those that do describe it below as the byte "at `pc + 1`".

In the table below `rA` means "the value stored in register number `a`" and `rB` means "the value stored in register number `b`."

| `icode` | `b` | Behavior | add to `pc` |
|---|---|---|---|
| 0 | | `rA = rB` | 1 |
| 1 | | `rA += rB` | 1 |
| 2 | | `rA &= rB` | 1 |
| 3 | | `rA` = read from memory at address `rB` | 1 |
| 4 | | write `rA` to memory at address `rB` | 1 |
| 5 | 0 | `rA = ~rA` | 1 |
| 5 | 1 | `rA = -rA` | 1 |
| 5 | 2 | `rA = !rA` | 1 |
| 5 | 3 | `rA = pc` | 1 |
| 6 | 0 | `rA` = read from memory at `pc + 1` | 2 |
| 6 | 1 | `rA +=` read from memory at `pc + 1` | 2 |
| 6 | 2 | `rA &=` read from memory at `pc + 1` | 2 |
| 6 | 3 | `rA` = read from memory at the address stored at `pc + 1` | 2 |
| 7 | | if `rA <= 0`, set `pc = rB` | N/A |
| 7 | | if `rA > 0`, do nothing | 1 |
| 0 | 0 | Decrement `rsp` and push the contents of `rA` on the stack | 2 |
| 0 | 1 | Pop the top value from the stack into `rA` and increment `rsp` | 2 |
| 0 | 2 | Push `pc + 2` onto the stack, set `pc = M[pc+1]` | 2 |
| 0 | 3 | `pc` = pop the top value from the stack | 2 |

Note the stack operations have the reserve bit set to 1. They are just not depicted here.