**Q1.**

Recall the following facts about the instructions on the ATmega32U2:

- All AVR instructions occupy 16 bits
- Three pairs of registers are used for addressing data items: X = r27:r26, Y = r29:r28, Z = r31:r30
- In AVR assembly language, each instruction is written as: **mnemonic destination, source**
  - *Mnemonics* are the abbreviated names of the instructions, e.g. `ldd`, `adiw`, etc.
  - The *source* may be a location or an instruction operand
- A *direct* address means that the address to operate on is specified in the instruction.
- An *indirect* address means that the address to operate on is specified in a register that the instruction will refer to.
- An *immediate* value is an operand contained in an instruction (in contrast to the operand being contained in a register).

The *Store Indirect From Register to Data Space using Index X* instruction stores the contents of a specified source register into the memory location addressed by the X registers (see https://www.microchip.com/webdoc/avrassembler/avrassembler.wb_ST.html):

| Syntax | Operands | Operation |
|---|---|---|
| `st X, Rr` | $0 \leq r \leq 31$ | (X) ← Rr |
| Program counter | Opcode | Flags |
| PC ← PC + 1 | 1001 001r rrrr 1100 | None |

The *Subtract Immediate* instruction subtracts a constant value from the specified register (see https://www.microchip.com/webdoc/avrassembler/avrassembler.wb_SUBI.html):

| Syntax | Operands | Operation |
|---|---|---|
| `subi Rd, K` | $16 \leq d \leq 31, 0 \leq K \leq 255$ | Rd ← Rd - K |
| Program counter | Opcode | Flags |
| PC ← PC + 1 | 0101 KKKK dddd KKKK | H, S, V, N, Z, C |

When answering the following questions, assume that the state of the registers before the instructions are executed is:

- `R14` contains the value `5`
- `R15` contains the value `7`
- `R16` contains the value `139`

- R17 contains the value 22
- The X registers (r27:r26) contain the value 6
- The Y registers (r29:r28) contain the value 8
- The Z registers (r31:r30) contain the value 10
- The PC register contains the value 25

(a) What is the binary op-code for the instruction `st X, R15`?

(b) If the data memory looks like this

| *Address* | *Contents* |
|---|---|
| ... | ... |
| 0x0004 | 10 |
| 0x0005 | 11 |
| 0x0006 | 12 |
| 0x0007 | 13 |
| 0x0008 | 14 |
| ... | ... |

before executing `st X, R15`, what does it look like *after* the instruction has been executed?

(c) What value is contained in R16 after executing the instruction `subi R16, 137`?

(d) After executing *both* instructions, what will the value of PC be?

**Q2.**

Fig. 1 is a simplified circuit diagram for a GPIO pin on the ATmega32U2 microcontroller.

Figure 12-2. General Digital I/O(1)

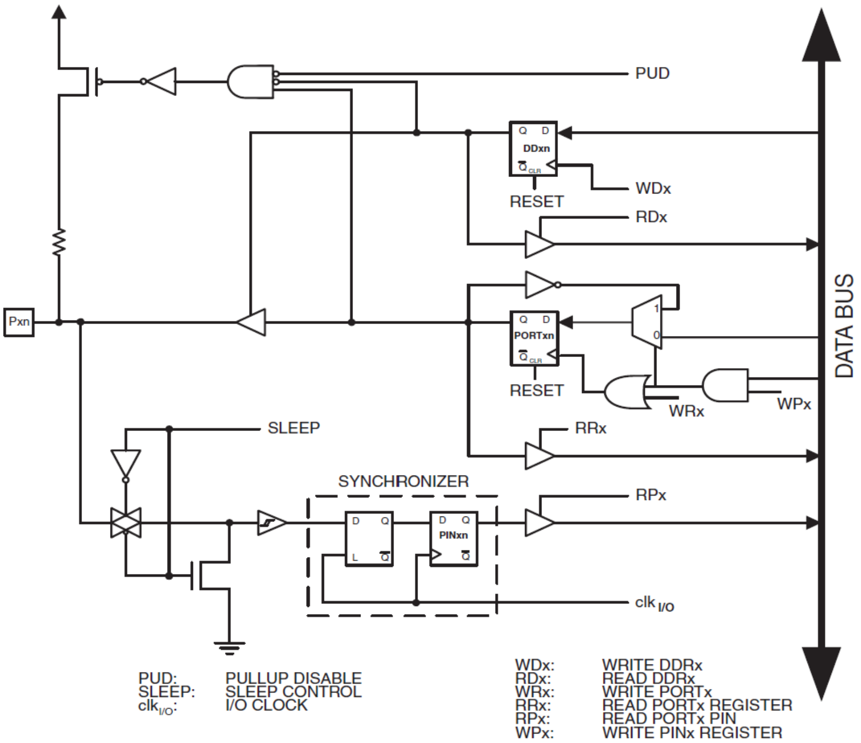

Figure 1: The logical circuit for one GPIO pin on the ATmega32U2 MCU.

(a) The `PUD` signal is a "Pull-Up Disable". Briefly explain why setting PUD to 1 would disable the pull-up resistor connected to *Pxn*.

(b) Under what conditions would the tristate buffer between *Pxn* and the *PORTxn* flip-flop be turned ON?

(c) What is the role played by the 2:1 multiplexer that provides the input to the *PORTxn* flip-flop?

**Q3.**

The following program for the MicroC simulator executes a simple function call:

```
load_immed r0, 0xA
load_immed r1, 0xB
call 0x05
out port, r7  ; Print the result
halt ; All done
push r0
push r1
add r0, r1
move r7, r0
pop r0
pop r0
ret
```

(a) Sketch a diagram showing the contents of the stack at the point in the program when the `add` instruction is executed. Your diagram should show the values in the stack, the addresses of those values, and the location of the stack pointer.

(b) Modify the program so that the body of the function iteratively adds `r1` to `r0` until the sum is greater than or equal to 100. What sum is output when you run this program?

**END OF QUESTIONS**