CSC 230
Fall 2022 (CRN 10817)
Midterm #1: Thursday, 6 October 2022

Marking guide

Students must check the number of pages in this examination paper before beginning to write, and report any discrepancy immediately.

- **All answers are to be written on this exam paper.**
- The exam is closed book. Other than the AVR reference booklet provided to you, no books or notes are permitted.
- When answering questions, please do not detach any exam pages!
- ***A basic calculator is permitted***. Cellphones must be turned off.
- Partial marks are available for the questions in sections B and C.
- The total marks for this exam is 76.
- There are ten (10) printed pages in this document, including this cover page.
- We strongly recommend you read the entire exam through from beginning to end before starting on your answers.
- **Please have your UVic ID card available for inspection by an exam invigilator.**

Section A (36 marks): For each question in this section, place an X beside all answers that apply. Each question is worth three (3) marks. *Partial marks are not given for incomplete answers*.

Remember: Hexadecimal numbers begin with `0x`; binary numbers begin with `0b`; octal numbers begin with `0`; and all other numbers are decimal. *All numbers are unsigned unless the question specifies otherwise.*

Question 1: The unsigned integer `0x725` can be represented as:

____ `0b10100100111`

____ `0b0111010101`

____ `0b11100100101`

____ `03445`

____ None of the above.

Question 2: The unsigned integer `0b11010101` is equivalent to:

____ `173`

____ `0xD5`

____ `0173`

____ `0x5D`

____ None of the above.

Question 3: The unsigned integer `0572` is equivalent to:

____ `0b000101111010`

____ `378`

____ `0x17A`

____ `0b000101111010`

____ None of the above.

Question 4: The decimal number `-101` represented as a 10-bit two's complement number is equivalent to:

____ `0b1000011011`

____ `0b1110011010`

____ `0b1001100101`

____ `0b1110011011`

____ None of the above.

Question 5: The seven-bit two's complement number `0b1010110` is equivalent to:

____ `0b011010110` as an nine-bit two's complement number.

____ `42`

____ `-26`

____ `-29`

____ None of the above.

Question 6: If we consider I/O port D, then:

____ we set the data-direction for each bit (input or output) by an appropriate value stored in the PIND and PORTD registers.

____ we set the data-direction for each bit (input or output) by an appropriate value stored in the DDRD register.

____ we read values from the port by reading from the PDDD register.

____ we read values form the port by reading from the POUT register.

____ None of the above.

Question 7: The lowest negative value that can be represented using a six-bit two's complement number is:

____ `0b100000`

____ `0b011111`

____ `-64`

____ `-31`

____ None of the above.

Question 8: The most positive value that can be represented using a seven-bit two's complement number is:

____ `0b1000000`

____ `0b0111111`

____ `64`

____ `63`

____ None of the above.

Question 9: The *fetch-decode-execute cycle*:

____ Describes the steps taken when the assembler generates a sequence of `fet`, `dec`, and `exe` instructions that appear in a loop.

____ Permits the AVR mega2560 to fetch and decode an instruction in one cycle, and to execute it in the next cycle(s).

____ Permits the AVR mega2560 to read the instruction from data memory, and then write the results to program memory.

____ Is another name for the meaning of the `fde` instruction .

____ None of the above.

Question 10: A *big-endian* computer system:

____ means that the system always addresses memory as *quadwords*.

____ stores the more-significant bytes of a 32-bit integer at lower addresses than the less-significant bytes.

____ is another way of saying that a system uses the von Neumann (i.e., Princeton) machine architecture.

____ stores bytes for a 32-bit integer in an order reversed to that of a *little-endian* system.

____ None of the above.

Question 11: The AVR architecture's *pseudo-registers* X, Y, and Z:

____ actually refer to specific pairs of general-purpose registers.

____ actually refer to memory-addressed I/O ports.

____ can be allocated as needed to any pair of registers (e.g. X to same as, say, r13:r12, for some assembly-language program).

____ may be used to hold unsigned 16-bit integers, signed 16-bit integers, or 16-bit memory addresses.

____ None of the above.

Question 12: The *Z flag* within the AVR status register:

____ can be directly set with the status register via the `ser` instruction.

____ is used by some branch instructions.

____ is used by the `rjmp` instruction.

____ is set as a result of using the `clr` instruction.

____ None of the above.

Section B (20 marks): One question

Question 13: Consider the following program written in AVR assembler:

```
.cseg
.org 0

start:      ldi r16, 0b00010111
            clr r17
labelZ:     add r16, r16
            dec r18
            inc r17
            cpi r17, 0x03
            brne labelZ
spin:       rjmp spin
```

a) How many executions occur for the `inc r17` instruction? *Explain your answer.* ***[4 marks]***

It is incremented three times – moving from 0 up to and including 3 (0 → 1, 1 → 2, 2→ 3. Once `cpi r17, 0x03` is encountered with `r17` equal to 3, the `brne` is not longer taken.

b) What is the value in `r16` when the program reaches the instruction `rjmp stop`? *Explain your answer, showing both hexadecimal and 8-bit twos-complement.* ***[10 marks]***

The initial value of r16 is 23; each time through the labelZ loop, r16 is added to itself – so first time through r16 goes from 23 to 46; second time through is goes from 46 to 92; and last time through it goes from 92 to 184.

Final value is 184. which is 0xB8. As 0xB8 is 0b10111000, using the sign rule produces 0b01001000 – which means the value is equivalent to -72 in eight-bit two's complement.

c) How would you modify the end of this program such that the value in `r16` would be stored at the highest address in SRAM? Answer this by providing the AVR assembler lines needed before or after (or both!) the `rjmp stop` line. *Explain your answer, and clearly describe any assumptions you are making.* ***[6 marks]***

The highest location is RAM is 0x21ff (or equivalently, RAMEND)

Therefore replace the last line of the code on the previous page with:

```
        ldi r31, HIGH(RAMEND)
        ldi r30, LOW(RAMEND)
        st Z, r16
stop:   rjmp stop     ; Yes, many of you noticed the typo
                      ; confusing spin & stop
```

(There were many possible correct answers.)

Section C (20 marks): Two questions

Question 14: 10 marks

Consider the following instruction:

```
mov r29, r14
```

Using the information provided to you in the “AVR Architecture Reference Booklet” that comes with this exam, what is the encoding of this instruction?

Your answer must not only show all work but also explain how you arrived at your answer.

The manual page for `mov` indicates the following encoding:

`0010 11rd dddd rrrr`

Where Rd is the destination register (here `r29`) and Rr is the source register (here `r14`). The five-bit binary for 29 is `0b11101`, and the five-bit binary for 14 is `0b01110`.

So if we begin by putting down the bits for r29, we have:

`0010 11r`**`1 1101`**` rrrr`

and then follow that with the bits for r14, we have:

`0010 11`**`0`**`1 1101 `**`1110`**

which gives us our encoding (i.e. **`0x2dde`**).

Question 15: 10 marks

Consider the following encoding of an instruction:

```
0x9513
```

Using the information provided to you in the "AVR Architecture Reference Booklet" that comes with this exam, what is the actual AVR instruction (i.e. mnemonic plus argument)?

Your answer must not only show all work but also explain how you arrived at your answer.

The first step is to view the binary representation of the 16-bit opcode

```
0b1001 0101 0001 0011
```

So which instruction is it? There are three instructions in the AVR booklet that begin with `0b1001`: `DEC`, `EICALL`, and `INC`. We can use a process of elimination to remove EICALL from consideration as its last four bits are `0b1001` – which is certainly not in the bit sequence we've written.

Is it `INC` or `DEC`? Again we can look again at the last four bytes of both INC and DEC, and we see it is the former with `0b0011` at the end (like the bit sequence we've written).

INC has the format:

```
0b1001 010d dddd 0011
```

and re-writing our sequence (i.e. bolding or underlining the necessary bits) gives us:

`0b1001 010`**`1 0001`**` 0011`

So it appears that the register number `0b10001` is 17, which means the encoded instruction is:

```
INC r17
```

(This page intentionally left blank.)