

# CISP317 Practice Final Exam

Prof. Tak Auyeung

December 9, 2003

**Instructions:** You may bring any material that is handwritten or printed *prior* to the examination to help you. You can also bring a calculator if you think it may help you. However, you can only use the calculator for numerical computations only. You *cannot* let your calculator compile a program, or to communicate with others.

You, as an individual, are expected to do your own work. This means you cannot seek, receive or otherwise acquire any assistance except clarifications from the professor during an examination. Any communication involving the contents of the subject matter or the examination is considered cheating. Do not initiate or accept such communication, or the result of your examination is automatically voided.

*New rules, read this!* As of 2003.09.22, I no longer deduct points for wrong answers. Each correct answer is worth one point, each wrong answer is worth zero point, and each unanswered question is also worth zero point. This means you *should* guess and leave no question unanswered.

As a result, I also need to adjust the letter grade assignment break points. For your individual examination, “A” means at least 94%, “B” means at least 74%, “C” means at least 54%, “D” means at least 34% and “F” means below 34%. The break points for the final grade are now 26.83%, 48.5%, 70.17% and 91.83% as minimums for “D”, “C”, “B” and “A”, respectively.

Please note that this change does not affect your letter grade at all, it is just a number game to make some people feel better about guessing.

Make sure you write down your name on the upper right corner *first*, otherwise I cannot give points to anonymous students!

The baseline is XX, there are YY questions.

1 Choose values of `r0` and `r1` such that the following instruction results in the `N` flag being set.

`add r0,r1`

A `r0=0x20, r1=0x30`

B `r0=0x40, r1=0x30`

C `r0=0x20, r1=0xf0`

D `r0=0xa0, r1=0x30`

E `r0=0xe0, r1=0x30`

2 Assuming unsigned 8-bit variables ‘a’ and ‘b’ are already loaded into registers (two registers in `r16` to `r31`), choose the correct implementation of the following pseudocode:

**while** (`a < b`) or (`b > 50`) **do**

`b ← b - 1`

`a ← a + 1`

**end while**

A L0:

`cp a,b`

`brcc L1`

`cpi b,50`

`brcc L2`

L1:

`dec b`

`inc a`

`rjmp L0`

L2:

```

B L0:
    cp a,b
    brcs L1
    cpi b,50
    breq L2
    brcs L2
L1:
    dec b
    inc a
    rjmp L0
L2:
C L0:
    cp a,b
    brcs L1
    cpi b,50
    brcc L2
L1:
    dec b
    inc a
    rjmp L0
L2:
D L0:
    cp a,b
    brcs L1
    cpi b,50
    breq L2
    brcs L2
L1:
    dec b
    inc a
    rjmp L0
L2:
E L0:
    cp a,b
    brcc L1
    cpi b,50
    brne L1
    brcc L2
L1:
    dec b
    inc a
    rjmp L0
L2:

```

3 Assuming the following program starts from the beginning and ends at label `eop`. What should be the value of register `r0` at label `eop`?

```

    ; ... code to set up the stack pointer
    sub r0,r0
    rcall sub1
    rcall sub2
eop: nop

sub1:
    rcall sub2
    rcall sub2
    ret

```

```
sub2:
    inc r0
    ret
```

- A 0
- B 2
- C 3
- D 4
- E 6

4 Referring to the program in the previous question, what is the number of bytes required for the stack?

- A 0
- B 1
- C 2
- D 4
- E 6

5 Which instruction can replace the following one without change of behavior?

```
lds r16,1
```

- A ldi r16,1
- B sts 16,r1
- C sts 1,r16
- D mov r1,r16
- E ldi r1,16

6 Using the 4-bit signed representation, the value -3 is represented as the bit pattern 1101<sub>2</sub>. Sign extension is a process to widen the binary representation of numbers. If we want to sign extend the representation of -3 to use 8 bits, what will it look like?

- A 00001101<sub>2</sub>
- B 11010000<sub>2</sub>
- C 11111101<sub>2</sub>
- D 11011111<sub>2</sub>
- E 11110011<sub>2</sub>

7 It is important to check that the result of an operation is within the range of numbers that can be represented. When two unsigned values, stored in 8-bit representation are added, how do we check the result is valid? Assume the values are stored in `r0` and `r1`, and we use the following instruction to perform the addition. Choose a conditional branch to `invalid_result` if and only if the result is out of the range of `r0`.

```
add r0,r1
```

- A brmi invalid\_result ; N = 1
- B brcs invalid\_result ; C = 1
- C brlt invalid\_result ; V = 1
- D breq invalid\_result ; Z = 1
- E brne invalid\_result ; Z = 0

- 8 What do we know at the `nop` instruction in the following code? Assume the state of pin 2 of port A remains the same for at least the amount of time taken to execute 10 instructions. Assume pin 2 of port A is configured as input.

```

        ldi    r16,5
L0:
L1:  sbic   PINA,2
    rjmp   L1
L2:  sbis   PINA,2
    rjmp   L2
    dec   r16
    brne  L0
    nop

```

- A the code gets to `nop` without any transitions at pin 2 of port A
  - B the code arrives `nop` after a 0-1 transition at pin 2 of port A
  - C the code arrives `nop` after a 0-1-0-1-0-1-0-1-0-1 transition pattern at pin 2 of port A
  - D this code never gets to `nop` because it is an infinite loop, regardless of the state of pin 2 of port A
  - E the code arrives `nop` after a 1-0 transition at pin 2 of port A
- 9 Choose the pseudocode that matches the following assembly code.

```

L1:  cp    V1, V2
    breq  L2
    inc   V1
    dec   V2
    rjmp  L1
L2:  nop

```

- A **if**  $V1 \neq V2$  **then**  
 $V1 \leftarrow V1 + 1$   
 $V2 \leftarrow V2 - 1$   
**end if**
- B **if**  $V1 = V2$  **then**  
 $V1 \leftarrow V1 + 1$   
 $V2 \leftarrow V2 - 1$   
**end if**
- C **while**  $V1 = V2$  **do**  
 $V1 \leftarrow V1 + 1$   
 $V2 \leftarrow V2 - 1$   
**end while**
- D **while**  $V1 \neq V2$  **do**  
 $V1 \leftarrow V1 + 1$   
 $V2 \leftarrow V2 - 1$   
**end while**
- E **repeat**  
 $V1 \leftarrow V1 + 1$   
 $V2 \leftarrow V2 - 1$   
**until**  $V1 = V2$

- 10 What is loaded into `r16` after the following code?

```

ldi    r26,low(L1)
ldi    r27,high(L1)
ld     r16,X
subi   r16,6

```

- A the address of label L1
- B -6
- C the value at address L1
- D the value at address L1 minus 6
- E the least significant byte of address L1