Department of Computer Science Spring 2015 New Mexico State University

Final Exam

C S 273 Machine Programming and Organization 5/4/2015

This is a closed book exam except that you can use the AVR instruction tables. You are also allowed to use a calculator. The total number of points is 100. Please answer all questions.

NAME: _____ SCORE: _____

1. Answer the following questions regarding subroutine call and stack.

Line no.	Program memory address		Instruction
1:	0x0204		ldi R18, -10
2:	0x0205		ldi R19, 23
3:	0x0206		call add
4:	0x0208	here:	rjmp here
5:	0x0209	add:	add R18, R19
6:	0x020A		call sub
7:	0x020C		ret
8:	0x020D	sub:	subi R18, 15
9:	0x020E		ret

(5 points) (a) When line 5 (labeled by add) is executed, what are the first two entries on the top of the stack?

Solution:

(stack top position)
0x08
0x02
⋮

(5 points) (b) When line 8 (labeled by sub) is executed, what are the first four entries on the top of the stack?

Solution:

(stack top position)
0x0C
0x02
0x08
0x02
⋮

(10 points) 2. Describe three methods to exchange information between C and assembly code.

Solution: Exchange of information can be achieved through

1. using global variables
2. calling assembly routine from C using arguments, which are assigned to registers by function call protocol
3. returning a value from assembly to C code through registers

3. Analog-to-digital conversion. The AVR ATMega328p system supports 10-bit ADC conversion.

(5 points) (a) If the input voltage range is 0 to 500 mV (1 mV = 0.001 V), what should be the reference voltage to maximize ADC precision within the input voltage range?

Solution: 500mV.

(5 points) (b) If the reference voltage is 1000mV, what is the ADC result for the input of 500mV?

Solution: Step size = 1000 mV / 2^{10};

Output is thus 500 mV / (1000 mV / 2^{10}) = 512.

4. Assembly and reverse assembly.

(10 points) (a) Assemble the following code to binary.

Line no.	Assembly code	Address	Binary code (high byte – low byte)
1	.ORG 0x300		
2	LDI R18, 9		
3	LOOP: DEC R18		
4	BRNE LOOP		

Solution:

Line no.	Assembly code	Address	Binary code (high byte – low byte)
1	.ORG 0x300		
2	LDI R18, 9	0x300	1110-0000-0010-1001
3	LOOP: DEC R18	0x301	1001-0101-0010-1010
4	BRNE LOOP	0x302	1111-0111-1111-0001 (offset k= -2 = 0b1111110)

(10 points) (b) Reverse assemble the following binary code to readable instructions

Assembly code	Address	Binary code (high byte – low byte)
	0x500	1001-1011-0001-1101
	0x501	1001-0100-0000-1100
	0x502	0000-0101-0000-0000
	0x503	1001-0101-0000-1000

Solution:

Line no.	Assembly	code	Address	Binary code (high byte – low byte)
1		.ORG 0x500		
2	CHECK:	SBIS 0x03,5	0x500	1001-1011-0001-1101
3		JMP CHECK	0x501	1001-0100-0000-1100
			0x502	0000-0101-0000-0000
4		RET	0x503	1001-0101-0000-1000

5. Timer/Counter.

(2 points) (a) For an 8-bit timer working in the normal mode, if its TCNT value is initialized to 50, how many events will have the timer/counter counted at the time of TOV flag turning from 0 to 1?

Solution: 256 - 50 = 206.

(2 points) (b) For an 8-bit timer working in the CTC mode, if its OCR value is initialized to 50, how many events will have the timer/counter counted at the time of OCF flag turning from 0 to 1?

Solution: 50 + 1 = 51.

(6 points) (c) Use pseudocode in English to explain how to check the timer without blocking the program execution. This means the program should not wait exclusively for the timer to expire, but instead the program can do some other tasks before the timer expires (such as tracking the sun light by adjusting the solar panel position).

Solution:

```
repeat {

  read timer OCF or TOV flag (non-blocking)

  if expired, process the timer expiration

  perform other tasks

}
```

(10 points) 6. Program memory. Write a subroutine called Add to sum up all the bytes from 0x450 to 0x500 in the program memory and save the sum in R18. Your program should return a value of

$$R18 = 0 + 1 + 7 + 3 + 4 + (-2) + \cdots + 8 + 55$$

in the following example. You can use either `LPM Rd, Z` or `LPM Rd, Z+` instruction.

Address	High byte	Low byte
0x0450	0	1
0x0451	7	3
0x0452	4	-2
⋮	⋮	⋮
0x0500	8	55

Solution:

```
Add:
    LDI R31, hi8(0x0450 << 1)
    LDI R30, lo8(0x0450 << 1)
    LDI R18, 0
    LDI R20, 0x500-0x450+1    ; number of addresses 177
L:  LPM R19, Z+
    ADD R18, R19
    LPM R19, Z+
    ADD R18, R19
    DEC R20
    BRNE L
    RET
```

7. Error checksum.

(5 points) (a) Calculate the error checksum code for the following numbers: 0x35, 0x52, -0x24, 0x05.

Solution: 0x35 + 0x52 + (-0x24) + 0x05 = 0x68. The checksum is defined to be the two's complement of 0x68, which gives us 0x98.

(5 points) (b) If the following decimal numbers are received but their order was scrambled during transmission:

153, 165, 172, 22

and we would not know which number is the checksum. Are you still able to determine if you have evidence for data corruption? If yes, please determine if there is any sign of data corruption; if no, please explain why.

Solution: 153 + 165 + 172 + 22 = 512

After removal of multiples of 256, it is zero. Thus we do not have evidence for data corruption from the checksum.

8. Interrupt handling.

(5 points) (a) When a particular interrupt occurs among many possible interrupts, how does the processor know where to find the corresponding interrupt service routine among many such routines?

Solution:

The interrupt will make program counter point to an entry in the interrupt vector table determined by hardware wiring. That entry will contain instructions to execute the corresponding interrupt service routine.

(5 points) (b) Describe the advantages and disadvantages of handling events through polling versus interrupt.

Solution:

	Polling	**Interrupt**
Hardware	No special hardware	Need interrupt unit
Efficiency	Low	High
Real-time responsiveness	Slow	Fast

9. Floating point numbers. We use the following format for non-negative 8-bit floating point numbers:

$$mm.mmmmee$$

where $mm.mmmm$ represents mantissa and ee is the exponent.

For example, we can convert the floating number to decimal:

$$10.001101 = (1 \times 2^1 + 0 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}) \times 2^1 = 4.375$$

(5 points) (a) What is the smallest number this 8-bit floating number can represent?

Solution: When the six mantissa bits are zero, it gives the smallest number of zero.

(5 points) (b) What is the largest number this 8-bit floating number can represent?

Solution: The largest number is when all bits are 1:

$$11.111111 = (1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}) \times 2^3 = 31.5$$