KEY # CSC 230 AI made Practice Exam 1 Fall 2024
Practice Exam #

Students must check the number of pages in this examination paper before beginning to write, and report any discrepancy immediately.
Total marks: 75
Time allowed: 2 hours

Section A (30 marks): For each question in this section, place an X beside all answers that apply. Each question is worth 6 marks. Partial marks are not given for incomplete answers.

Question 1: The unsigned integer 0x8F4 can be represented as:
_X__ 0b100011110100
___ 0b111110100
_X__ 02364
___ 0x4F8
___ None of the above

Question 2: When using the DDRD register for Port D configuration:
_X_ Setting a bit to 1 configures the corresponding pin as an output
_X__ Reading PIND gives the current state of Port D pins
_X__ Writing to PORTD affects pull-up resistors when pin is configured as input
_X__ The register must be configured before using PORTD
___ None of the above

Question 3: Regarding the AVR stack operations:
_X__ The stack grows toward lower memory addresses
___ push decrements SP after storing a value
___ pop increments SP after retrieving a value
___ SP points to the next free location
___ None of the above

Question 4: In the context of AVR's 8-bit arithmetic:
_X__ The carry flag is set when there's a carry from bit 7
_X__ add r16,r16 performs multiplication by 2
_X__ dec sets the Z flag if the result is zero
_X__ adc adds the carry flag to the result
___ None of the above

Question 5: For the AVR instruction set:

_X__ ldi can only be used with registers r16 to r31
___ mov can copy data between any registers
_X__ sts can store values to any data memory location
_X__ cp sets the carry flag if the second operand is greater
___ None of the above

Section B (25 marks): Short Answer Questions

Question 6 (5 marks):
Explain how the Z flag in the AVR status register is affected by the CP instruction. Include an example showing specific register values.

The CP instruction sets the Z flag when the comparison result is zero (i.e., when the two operands are equal). For example:

```
ldi r16, 0x45      ; Load first value
ldi r17, 0x45      ; Load second value
cp r16, r17        ; Compare r16 - r17
                   ; Z flag will be set because 0x45 - 0x45 = 0
```

The Z flag can then be used with conditional branches like BREQ to test for equality.

Question 7 (5 marks):
Describe the purpose and function of the stack pointer registers SPH and SPL in the AVR architecture. Why are two registers needed?

SPH and SPL form a 16-bit stack pointer in the AVR architecture:
- Two registers are needed because AVR has a 16-bit address space but 8-bit registers
- SPH holds the high byte (bits 15-8) of the stack pointer
- SPL holds the low byte (bits 7-0)
- Together they can address the full 64KB of memory
- The stack pointer must be initialized to RAMEND at program start

Question 8 (5 marks):
Compare and contrast the breq and brne instructions in terms of their operation and typical usage scenarios in AVR assembly programs.

BREQ (Branch if Equal) and BRNE (Branch if Not Equal) are complementary branch instructions:
- BREQ branches if Z flag is set (result was zero/equal)

- BRNE branches if Z flag is clear (result was non-zero/not equal)
- Typically used after comparison instructions (CP, CPI)
- BREQ commonly used for loop termination conditions
- BRNE commonly used for loop continuation conditions

Question 9 (5 marks):
Explain how the AVR architecture handles signed versus unsigned arithmetic operations. Use specific instructions as examples.

AVR handles signed vs unsigned arithmetic through:
- Separate signed (SBRC) and unsigned (BRCS) conditional branches
- Different status flags:
- V (overflow) for signed operations
- C (carry) for unsigned operations
- Instructions like ADD work the same for both, but interpretation differs
- Special instructions exist for signed comparisons (e.g., BRLT vs BRLO)
- Two's complement representation used for signed numbers

Question 10 (5 marks):
Describe the relationship between RAMEND and stack initialization in AVR programs. Why is proper stack initialization important?

RAMEND and stack initialization relationship:
- RAMEND is the highest address in RAM
- Stack should be initialized to RAMEND for optimal memory usage
- Proper initialization prevents stack overflow into program variables
- Stack grows downward from RAMEND
- Without proper initialization, stack operations may corrupt memory

Section C (20 marks): Programming Problems

Question 11 (8 marks):
Write an AVR assembly function that takes two 8-bit values passed in r16 and r17, and returns their greatest common divisor in r16. Show your complete solution including comments.

```
; GCD function using Euclidean algorithm
; Input: r16, r17
; Output: r16 contains GCD

gcd:
   push r18            ; Save working register

gcd_loop:
   mov r18, r16        ; Save r16
   mov r16, r17        ; r16 = r17
   clr r17             ; Clear r17 for division

 div_loop:
      cp r18, r16              ; Compare remainder with divisor
      brlo div_done            ; If remainder < divisor, division done
      sub r18, r16             ; Subtract divisor from remainder
      inc r17                  ; Increment quotient
      rjmp div_loop            ; Continue division

 div_done:
   mov r17, r18               ; Move remainder to r17
   tst r17                    ; Test if remainder is zero
   brne gcd_loop              ; If not, continue GCD calculation
   pop r18                    ; Restore working register
   ret                        ; Return with GCD in r16
```

Removed Question 12

Question 13 (4 marks):

Write the AVR assembly code to implement the following C statement:
if(x > 5 && x < 10) y++; // Assume x is in r16 and y is in r17

```
cpi r16, 6        ; Compare x > 5
  brlo done       ; Branch if less (not greater)
  cpi r16, 10     ; Compare x < 10
  brsh done       ; Branch if same or higher
  inc r17         ; Increment y
done:             ; End of if statement
```

Question 14 (2 marks):
Given a string stored in program memory starting at label MESSAGE, write the code to copy it to data memory location 0x200.

```
  ldi ZH, high(MESSAGE<<1) ; Load Z pointer with program memory address
  ldi ZL, low(MESSAGE<<1)
  ldi XH, high(0x200)    ; Load X pointer with data memory address
  ldi XL, low(0x200)

copy_loop:
  lpm r16, Z+           ; Load byte from program memory
  st X+, r16            ; Store to data memory
  tst r16               ; Check for null terminator
  brne copy_loop        ; Continue if not end of string
```

Question 15 (2 marks):
Write the code to configure Timer0 in CTC mode with a prescaler of 64.

```
  ldi r16, (1<<WGM01)    ; CTC mode
  out TCCR0A, r16
  ldi r16, (1<<CS01)|(1<<CS00)    ; Prescaler 64
  out TCCR0B, r16
```