ECE220 Midterm 1 Review

Author: Members of HKN

Fall 2024

1 Memory-Mapped I/O

(a) What is the difference between a callee-saved and a caller-saved register?
Answer: A callee-saved register is saved inside the subroutine (hence callee, as the subroutine is what is BEING called), then restored JUST before the subroutine finishes. A caller-saved register is saved JUST before the JSR call to the subroutine (hence caller, as the routine DOING the call to the subroutine saves the registers), then restored AFTER the call.

(b) What is the difference between polling I/O vs. Interrupt-Driven I/O?
Answer: Polling is when the routine constantly checks the status of the device and "spins" until it reads a ready status from the device. Interrupt-driven I/O is when the routine executes its intended functionality until a device with higher priority "interrupts" the routine. Control is handed over to the processor to allow an interrupt handler to execute.

(c) Fill in the blanks of the following statements relating to Memory-Mapped I/O.

Certain device registers are mapped to certain memory locations. However, the registers physically are separate from the memory. Memory-mapped device registers are a common way to interface computer systems with devices.

2 Traps and Subroutines

(a) Why would we need to have service routines (known as TRAPs)? Name three reasons.
Some sufficient answers:

i. Implementation for a service routine can be implemented once and called multiple times if needed.

ii. Hide system-specific implementation from programmers.

iii. Protect system resources from reckless/careless programming practices.

iv. Provide a wrapper to interface with devices I/O when reading/writing to devices is difficult.

(b) **Shifting :(**

Write a subroutine that performs a logical left shift of R1 by the value stored in R2, and reports the result in R1. You may assume that R2 contains a positive number. Only R6 and R1 can be modified/read from, and only R2 can be read from.

```
.ORIG x3100
SHIFTL
AND R6, R6, #0 ; zero out R6 to be used as decreasing loop counter
ADD R6, R6, R2 ; store R2 at R6
LOOP BRz DONE
ADD R1, R1, R1 ; Left-shift by 1
ADD R6, R6, #-1
BRnzp LOOP
DONE
RET
.END
```

(c) **Permute Quarters**:

A value stored in a 16-bit LC3 register can be divided into four equal parts of four bits each:

$$X = X_1 \ X_2 \ X_3 \ X_4$$

Write a subroutine, PERMUTE, that reorders R5 as follows:

$$\text{R5} = X_1 \ X_3 \ X_2 \ X_4$$

Assume all registers are caller-saved. You should not use loops. You may assume the existence of a SHIFTR subroutine, which right-shifts R1 the number of times given by R2 as expected, no other register is required. [Hint: the previous subroutine might be useful]

;assume the previous subroutine has been implemented correctly at other locations
;all registers values are caller-saved, so we are free to use any register except R5, but remember that those values will be overwritten in later operations

```
.ORIG x3100
; assume code for entry point is omitted

PERMUTE
; write the code here
ST R7, SAVER7 ; save R7
LD R1, X3
LD R2, FOUR
LD R3, X2
LD R4, CLEARX2X3
AND R1, R5, R1 ; store X3 of R5 at R1
AND R3, R5, R3 ; store X2 of R5 at R3
AND R5, R5, R4 ; zero out middle of R5
JSR SHIFTL ; left shift R1(X3) by 4
ADD R5, R5, R1 ; add left shifted X3 to R5
ADD R1, R3, #0 ; load R3(X2) to R1
JSR SHIFTR ; right shift R1(X2) by 4
ADD R5, R5, R1 ; add right shifted X2 to R5
LD R7, SAVER7 ; load previous R7 value
; code ends here
RET
; code omitted
;
HALT ; labels below
CLEARX2X3 .FILL 0xF00F
FOUR .FILL #4
```

```
X2 .FILL 0x0F00
X3 .FILL 0x00F0
SAVER7 .FILL #0
.END
```

(d) **Trap Concepts**

Given the figures below, determine the memory address of the TRAP vector table that will be accessed and which TRAP service routine will be executed.

User Program ASM Code
;
;
TRAP 0xAA
;
;

Address	Value
0xAA	0x05C0
0x1AA	0x05D0
0x2AA	0x05E0
0x3AA	0x05F0
...	
0x05C0	Routine α
0x05D0	Routine ω
0x35C0	Routine κ
0x35D0	Routine π

Trap Vector Table Entry:

a. 0xAA
b. 0x1AA
c. 0x2AA
d. 0x3AA

Answer: 0xAA

Trap Routine Executed:

a. α
b. ω
c. κ
d. π

Answer: α

Of the following steps executed during a TRAP, in what order are they executed?

a. Return to User Program
b. Execute Trap Routine
c. Access Trap Vector Table

1) Access Trap Vector Table
2) Execute Trap Routine
3) Return to User Program

3 Stack Operations

(a) Given the following input sequence of numbers: "24609846117", write a sequence of pushes and pops that produces this output: "64098116472".

Answer: PUSH PUSH PUSH POP POP PUSH POP PUSH POP PUSH POP PUSH PUSH PUSH PUSH POP POP POP POP PUSH POP POP

(b) Two Parts:

i. Write the expression $(((4 * 2) + 1)/3) + 5$ in postfix notation.
Answer: 142*+3/5+

ii. Write the following postfix expressions in mathematical notation and indicate what they evaluate to (if they are not valid, write "not valid")

* 6721*-5/*
  Answer: 6 * ((7 - (2 * 1)) / 5), evaluates to 6
* 83+632-+1*+-
  Answer: not valid
* 2233*2+5-+8/-
  Answer: 2 - (((2 + (3 * 3)) - 5) + 2)/ 8, evaluates to 1
* 89+6-44
  Answer: not valid

(c) **MP2 Postfix Calculator:** This sequence is input to the console: 445+3/8*-= Draw the stack (and where the stack pointer points to) after:

i. 5 has been input
ii. + has been input
iii. * has been input
iv. = has been input

Assume that the stack pointer points to the address **one above the most recent-pushed entry.** Remember that a POP does NOT remove an item from memory but simply changes the stack pointer!

Answers:

	⟵
5	
4	
4	

5	⟵
9	
4	

8	⟵
24	
4	

8	
24	⟵
-20	

4 C Programming

(a) What will be the output of the following C Program?

```c
int main() {
  int i;
  for (i = 3; i < 13; i ++ )
  {
    if (i % 3 == 1)
    {
      printf("Bong\n");
              }
    if (i % 2 == 0)
    {
      printf("Ding\n");
      continue;
    }
    printf("Odd\n");

  }
  return 0;
}

```

Answer:
Odd...
Bong
Ding
Odd...
Ding
Bong
Odd...
Ding
Odd...
Bong
Ding
Odd...
Ding

(b) What is the return value of this program?

```c
int foo(int x, int y);

int main()
{
  int x = 3;
  int y = 4;

  x = y + foo(x,y);
  y = x - foo(x,y);

  return x + y;



}

int foo(int x, int y)
{
  int a = x + y;
  int b= x - y;
  a += x--;
  y ++;
  a += (y + 1);
  return b + a + --x;


}


```

Answer: -44

5 Conceptual Questions

(a) What is the order of access for a stack abstract data type?

Answer: LIFO (Last-In-First-Out) or FILO (First-In-Last-Out)

(b) Define overflow and underflow.

Answer: Overflow: Attempting to push when stack is full. Underflow: Attempting to pop when stack is empty

(c) True or False. Please explain your answer.

i. Interrupts are more efficient than polling.

Answer: True. We dont need to waste resources constantly checking a bit, we can just have a signal that tells us when to interrupt.

ii. There are up to 8 possible TRAP service routines.

Answer: False. There are many more.

iii. TRAPs shield programmers from system specific details.

Answer: True. We dont need to know how traps stop the system, just that they do.

iv. PSR and PC are pushed to the User stack before executing an interrupt service routine.

Answer: False. PSR (Program Status Register) and PC are pushed to the supervisor stack, and are something the system does, and we are not allowed to interfere with.

v. An item is deleted after being pushed off the stack.

Answer: False. After being popped off the stack, the item is still in memory, the computer will simply overwrite it if something else is pushed on.

vi. TRAP service routines are provided as part of the system code.

Answer: True. Traps are implemented by the system, and are part of system code.