Midterm 1 Review

EECS 370 – Introduction to Computer Organization - Winter 2016

Profs. Valeria Bertacco & Reetu Das

EECS Department
University of Michigan in Ann Arbor, USA

Announcements

- No class on THURSDAY 2/11
- GSI/IA review – WEDNESDAY 2/10 @6pm BBB1670
- Special news for students attending their assigned lecture section…

Exam: Time

- Thursday February 11, 2016 – 7-8:30pm

- No Michigan time, exam starts at 7pm sharp
  - Show a few minutes early to your assigned room

- Closed book/notes
  - No phones, computers, calculators, smart watches, nothing electronic
  - You are only allowed
    a) #2 pencils,
    b) eraser and
    c) one letter-size single-sided sheet of paper with any notes you wish to have
- Bring your Mcard to the exam

Exam: Location

- Check your email for the location
  You should have received an email from either myself or Prof. Winsor with your exam location

- Go to your assigned room!
  Exams taken in the wrong room show up as missing from the correct room's roster.

Preparing for the exam

- Review sessions
  - Today's lecture - Profs
  - Wednesday 2/10 6 - 9pm – BBB1670 – GSIs/IAs
- Office Hours
  - Tuesday 2/9 – 10.30-noon – Prof. Austin
  - Tuesday 2/9 – 3pm-4.30pm – Prof. Bertacco
  - Tuesday 2/9 – 5.30-7pm - Jack
  - Wednesday 2/10 – 10-11.30am – Nathan and Olena
  - Wednesday 2/10 – 1.30-3pm - Olena
  - Wednesday 2/10 – 2-.3.30pm – Alan
  - Thursday 2/11 – 11.30am-1pm – Prof. Austin
  - Thursday 2/11 – 1-2.30pm – Prof. Bertacco
  - Thursday 2/11 – 3 – 6pm - Gabe

Midterm Material

Covers all material through Floating Point Arithmetic

- Covered in class – Lectures 1- 8 (slide 12)
- Related to Project 1
- Covered with Homeworks 1, 2 and Problem 1 of HW3

- Topics NOT covered
  - Combinational logic
  - Sequential logic
  - Adders
  - Finite State Machines

  Don't forget them! – they will be part of MIDTERM 2

Preparation strategies

- Studying tips
  - Go through the lecture notes, slide by slide
    - Do the class problems without looking at the answers
  - Skim through the book reading material
  - Re-solve homework problems

- Practice with old exams
  - Exams from the past 2 semesters with answers are posted
  - Great practice, get used to “exam-type” questions
  - Note: topics covered may be a bit different
    - e.g., MIPS instead of ARM, caller/callee more in depth
  - Create realistic exam setup: time yourself, don’t look at the solution until time is up, use your cheatsheet
    - Do the old tests without looking at the solutions!

Midterm Strategy

- Expect 7-10 problems covering all the material
  - Easier problems first, harder ones towards end
- Pace yourself
- Do not spend all your time on a single problem. If you don't understand a problem, move ahead and come back to it later
- For verbose questions you may want to read on a per-need basis. But do not assume to know what the text says!
- Give yourself a few minutes at the end to check your work.
- Most importantly, don't panic

Important Topics

- Representing values in hardware
  - Binary, octal, hexadecimal conversions
  - 2's complement representation
  - Floating point formats
  - Addition and multiplication in floating point
- Instruction sets
  - RISC vs CISC / load-store vs. memory
  - Assembly code ARM/ LC-2k / others: write & understand
  - Converting to machine code
  - Addressing modes for load/store instructions
  - Loading immediates in ARM
  - Conditional ARM assembly

Important Topics

- Converting C to assembly and back
  - Data alignment
  - Basic statements
  - Control flow constructs
- Running programs
  - Data organization (stack, heap, static)
  - Stack frames, stack and frame pointers
  - Object files: symbol table and relocation table
  - Caller/callee-saved registers
  - Compiler, linker, loader

What You Don't Need to Know

- No details of carry look-ahead bit computations
  - Just understand the benefits over ripple-carry and what P and G mean
- No history of INTEL processors

Select Questions From Old Exams

Just because it is not covered in this review it does not mean that it is not important!

Assembly – W03Q10 (adapted)

What does this sequence of LC2K instructions do? What sequence of two ARM instructions could be used to replace this sequence ?

```
add 1 1 2
add 2 2 2
add 2 1 2
add 2 2 2
```

ISA – W05Q1

Assume a 64-bit wide RISC machine with 64 registers and 200 distinct instructions. Its branch instruction compares two registers for equality, and uses the 2's complement offset field to compute the PC-relative target. What is the branch target range?

Addressing – F07 exam (adapted)

Consider the following ARM assembly program:

```
ldrb   r4, [r0,#100]
strh   r4, [r4,#-42]
ldrsh  r3, [r1,#2]
str    r3, [r0,#101]
```

Initial value for Register 1 is:
r1 = 0x0000 0064
r0 = 0

Fill in the tables.

Reg	Initial value	After inst 1	After inst 2	After inst 3	Final values
r3	0x0000 0020				
r4	0x0000 0074				

Address	Initial	After inst 1	After inst 2	After inst 3	Final values
100_{10}	0x91				
101_{10}	0x34				
102_{10}	0xC7				
103_{10}	0x84				
104_{10}	0x57				
105_{10}	0xB3				

Memory layout – F05 exam (adapted)

- How many bytes does the C data structure below require?

```
struct foo {
    char a;
    int b, c;
    char d[10];
};
```

- How can you rewrite the struct declaration so that it uses less memory ?

Linker – F05 exam

file1.c

```
1: int c;
2: void foo(int a) {
3: int b;
4: reference to b
5: reference to c
6: bar(a);
7: }
```

file2.c

```
8: void bar(int x) {
9: int y[5];
10: reference to y
11: reference to x
12: reference to c
13: }
```

- What symbols will appear in the symbol tables of the two files ?
- What line numbers will be referenced in the two relocation tables ?

Caller/Callee – W07 #8

```
int function1(int arg) {
  int x, y, w, z, k ;
  x = 15;
  y = 12;
  function2(x);
  z = x + y;
  k = z;
  while (y > 0) { // iterates 12 times
    function2(y-1);
    w = k – 2;
    y--;
  }
  return 0;
}
```

```
int function2(int a) {
   int b, c, d;
   b = a + 1;
   c = a / 2;
   d = b * c;
   printf("%d %d\n", a, c);
   d = d * d;
   return d;
}
```

The architecture has 3 caller, 3 callee registers

function1 : w and z are stored in callee-saved registers, while x, y and k are in caller-saved registers.

function2 : c and d are in callee-saved registers, while b is in a caller-saved register.

- Given the register mapping above, how many ldr/str must be included in the assembly code of these functions to handle the store and restore of registers?
- How many ldr/str must be executed when running this program?

Floating point - F07Q7

- AMD's new SSE5 extension to the x86 ISA includes a new 16 bits floating point datatype. The 16 bits of a number stored in this format are allocated as follows:
  - 1 bit sign
  - 5 bit exponent with a bias of 15
  - 10 bit mantissa (a.k.a. significand)
- All other aspects of this format are exactly the same as the standard IEEE floating point studied in class.

- a) Circle the numbers among those listed below that can be represented exactly using the SSE5 floating point format:

  2^{18} π 2^{-6} -2^{12} -2^{-12} 1/3

- b) What decimal number is represented by the following?

  1 10111 1001001010
  sign exponent mantissa

Conditional ARM assembly - F13Q18

Select the C code that computes the same function as the ARM assembly below

```
LOOP  cmp   r0, r1
      moveq r2, r3
      addne r2, r2, #1
      cmp   r2, #10
      blt   LOOP
```

A

```
do {
  if (r0 != r1){
      r2 = r3;
  } else {
      r2++;
  }
}while(r2 >= 10);
```

B

```
do {
  if (r0 == r1){
      r2++;
  } else {
      r2 = r3;
  }
}while(r2 < 10);
```

C

```
do {
  if (r0 == r1){
      r2 = r3;
  } else {
      r2++;
  }
}while(r2 < 10);
```

D

```
while (r2 < 10) {
  if (r0 == r1) {
      r2 = r3;
  } else {
      r2++;
  }
}
```

E

```
while (r2 < 10) {
  if (r0 == r1) {
      r2++;
  } else {
      r2 = r3;
  }
}
```

Linker - Q23F13

Consider the following C program file `fun.c`:

```
        int *a;
        int h = 0;
        struct {
             int x;
             double y;
        }S;
        int foo(int z) {
1.           int *i = a;
2.           h++;
3.           z = *i + 1;
4.           z = bar(10);
5.           return h+z;
        }
```

1. What is the complete set of symbols present in the symbol table of fun.o?

2. What is the complete set of instruction's locations present in the relocation table of fun.o?

Caller/callee – F11Q9

You are assigned the task of compiling the function below:

```
void mystery_func() {
  int a = 3, b = 4, c = 5, i = 6;
  if (b > 100) {
    return i;
  }

  for (i = 0; i < 10; i++) {
    c = a + i;
    c = another_func(a, b * i);
    printf("Iteration %d: %d %d %d\n", i, a, b, c);
    a = i * i;
  }

  return a;
}
```

Assume that `another_func` is a user-defined function located elsewhere.

Caller/callee – F11Q9 – cont.

1. When we say that register values are stored in/restored from memory by following either the caller-save or the callee-save convention, where in memory do these values usually reside in?

2. Next, identify how many store and load operations are **executed** for each variable when mystery_func() is executed **once**, depending on whether the variable is mapped to a caller-saved vs. a callee-saved register.

3. Finally, assuming that we have 1 caller-saved and 3 callee-saved registers to work on, determine the best assignment (to minimize the total number of load/store operations) for each variable.