

Intro. to Computer Architecture
CSE 240 Autumn 2004

Homework 4 Solutions
DUE: Mon. 11 October 2004

Write your answers on these pages. Additional pages may be attached (with staple) if necessary. Please ensure that your answers are legible. Please show your work. Due at the *beginning of class*. Total points: 60.

1. [6 Points] **Instruction Encoding.** Suppose a machine encodes instructions in 32-bits according to the following format. Assume there are 290 opcodes and 60 registers.

| | | | | |
|--------|----|-----|-----|--------|
| OPCODE | DR | SR1 | SR2 | UNUSED |
|--------|----|-----|-----|--------|

- (a) What is the minimum number of bits required to represent the OPCODE field?

Answer: Eight bits can represent 256 values ($2^8 = 256$), so (in this case) eight isn't enough. Nine bits can represent 512 values ($2^9 = 512$), so we need nine bits to represent 290 values. Unfortunately, we are not making efficient use of these bits. If possible, the ISA designer might want to find a way to get the number of opcodes down to 256, so that opcode may be represented in one fewer bits.

- (b) What is the minimum number of bits required to represent each of the register fields (e.g., DR)?

Answer: First, it is important to observe that all register fields (i.e., DR, SR1, and SR2) can name all registers, so they will all be the same size. Five bits isn't enough, because $2^5 = 32 < 60$. Six bits is just enough, because $2^6 = 64 > 60$. Heck, why not give this machine 64 registers? It won't increase the encoding space.

- (c) What is the maximum number of bits left available for the UNUSED field and how many values could it encode?

Answer: We know the whole instruction is 32 bits and the opcode consumes nine bits and each register field consumes six bits (for a total of $6 \times 3 = 18$ bits). We are then left with $32 - 9 - 18 = 5$ bits for the immediate field. In five bits we can represent $2^5 = 32$ different values.

2. [12 Points] **LC-3 Instruction Encoding.** For these questions assume the LC-3 instruction encoding (inside the back cover of your textbook). You may also need to consult Appendix A.

- (a) What is the range of values (in decimal) that may be specified by the immediate field in an AND instruction?

Answer: The immediate field in an AND instruction is five bits. This immediate represents a 2's complement number, so integer values from -16 to 15 may be represented.

- (b) What is the range of values (in decimal) that may be specified by the PCoffset field in a BR instruction?

Answer: The PCoffset field in a BR instruction is a 9 bit 2's complement number. Therefore, it may represent integer values from -256 to 255.

- (c) What is the range of values (in decimal) that may be specified by the offset field in an LDR instruction?

Answer: The offset field in an LDR instruction is a 6 bit 2's complement number. Therefore, it may represent integer values from -32 to 31.

- (d) What is the relationship between JMP and RET? In particular, why do they have the same bits in the opcode field?

Answer: Both instructions impact control flow by updating the PC. JMP places the contents of any register into the PC, while RET places the contents of R7 in the PC. In fact, if you examine the instruction encodings of both (p. 529) you will see that RET is encoded just like a JMP except the BaseR field is set to R7. From this we can conclude that there is not a special RET machine instruction. Rather, it is a special case of JMP.

- (e) Give the encoding of two LC-3 instructions that together increment register R3 by 20. Complete the following table.

Answer: Only five bits are available to encode an immediate operand in an ADD instruction. With a 5-bit 2's complement value we can represent values from -16 to 15, so we can not increment by 20 in one instruction. Instead, we can increment by 15 in the first instruction and 5 in the next.

| Address | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Instruction |
|---------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-------------------------|
| x3001 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | R3 <- R3 + 15 |
| x3002 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | R3 <- R3 + 5 |

3. [14 Points] **LC-3.** Suppose you want to execute the program made up of the instructions in the final column of following table.

| Address | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Instruction |
|---------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---------------|
| x3001 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | R2 <- M[R1+0] |
| x3002 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | R3 <- M[R1+1] |
| x3003 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | BRzp x3006 |
| x3004 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | R3 <- NOT R3 |
| x3005 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | R3 <- R3 + 1 |
| x3006 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | R2 <- R2 + R3 |
| x3007 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | M[R1+2] <- R2 |

- (a) Give the binary encoding of each instruction in the table. Write your answers in the table, above.

Answer: See table, above.

- (b) Trace the execution of the above program, starting at x3001, by completing the following table. Give the PC and instruction to execute in the first two columns, and give the state of the registers and condition codes *after* the execution of that instruction (**leave an entry blank if it has not been changed by the instruction**). The initial register state and the effect of the first instruction are given in the first two rows. Assume memory locations x3100 and x3101 contain 2 and -3, respectively.

Answer:

| PC | Instruction | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | CCs |
|---------------------------------|-------------------------|----|-------|----------|-----------|----|----|----|----|----------|
| <i>initial register state</i> ⇒ | | 0 | x3100 | 0 | 3 | 4 | 5 | 6 | 7 | |
| x3001 | R2 <- M[R1+0] | | | 2 | | | | | | P |
| x3002 | R3 <- M[R1+1] | | | | -3 | | | | | N |
| x3003 | BRzp x3006 | | | | | | | | | |
| x3004 | R3 <- NOT R3 | | | | 2 | | | | | P |
| x3005 | R3 <- R3 + 1 | | | | 3 | | | | | |
| x3006 | R2 <- R2 + R3 | | | 5 | | | | | | |
| x3007 | M[R1+2] <- R2 | | | | | | | | | |

- (c) In a sentence, what does this code compute?

Answer: This code computes the sum of M[R1+0] and the absolute value of M[R1+1].

4. [16 Points] **LC-3.** The following (bit-level) memory contents represent an LC-3 program.

| Address | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Instruction |
|---------|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|--------------------------|
| x3001 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | R1 <- M[R0+0] |
| x3002 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | R2 <- R2 AND 0 |
| x3003 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | R3 <- M[R0+1] |
| x3004 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | R2 <- R2 + R1 |
| x3005 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | R3 <- R3 + -1 |
| x3006 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | BRp x3004 |
| x3007 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | M[R0+2] <- R2 |

- (a) First, determine what each instruction does. Write this next to each instruction (above) in a manner similar to that of the previous problem.
- (b) Next, trace the execution of the above program, starting at x3001, by completing the following table. Give the PC and instruction to execute in the first two columns, and give the state of the registers and condition codes *after* the execution of that instruction (**leave an entry blank if it has not been changed by the instruction**). The initial register state and the effect of the first instruction are given in the first two rows. Assume memory locations x3100 and x3101 contain 3 and 2, respectively.

Answer:

| PC | Instruction | R0 | R1 | R2 | R3 | R4 | R5 | R6 | R7 | CCs |
|--------------|---------------------------------|-------|----|----------|----------|----|----|----|----|----------|
| | <i>initial register state</i> ⇒ | x3100 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| x3001 | R1 <- M[R0+0] | | 3 | | | | | | | P |
| x3002 | R2 <- R2 AND 0 | | | 0 | | | | | | Z |
| x3003 | R3 <- M[R0+1] | | | | 2 | | | | | P |
| x3004 | R2 <- R2 + R1 | | | 3 | | | | | | |
| x3005 | R3 <- R3 + -1 | | | | 1 | | | | | |
| x3006 | BRp x3004 | | | | | | | | | |
| x3004 | R2 <- R2 + R1 | | | 6 | | | | | | |
| x3005 | R3 <- R3 + -1 | | | | 0 | | | | | Z |
| x3006 | BRp x3004 | | | | | | | | | |
| x3007 | M[R0+2] <- R2 | | | | | | | | | |

- (c) Describe what this code does, assuming execution starts at address x3001. What registers or memory serve as input to this code? And what registers or memory serve as output? Be very careful in determining the input and output (*i.e.*, just because a register appears in the code does not mean that it is input or output).

Answer: This code performs multiplication. R0 is an input register containing the address of the input and output in memory. The memory at addresses R0+0 and R0+1 contain the two operands to multiply. The result is stored in memory at address R0+2.

- (d) Under what circumstances will this program fail to perform its principal task?

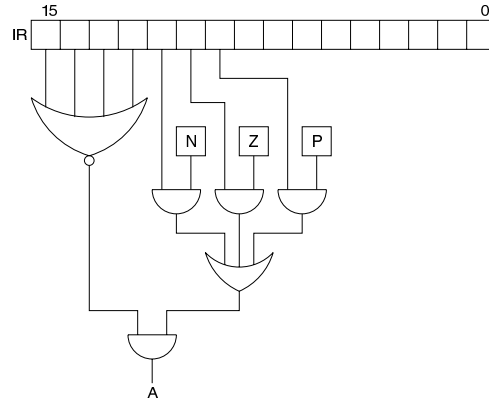
Answer: This program will fail to multiply when the second operand (at R0+1) is 0 or negative. Note that it *will* work when the first operand (at R0+0) is 0 or negative.

5. [6 Points] **Instruction Processing.** The PC, IR, MAR, MDR, and RF (register file) are structures written in various phases of the instruction processing cycle, depending on the opcode of the particular instruction being executed. In each cell in the table below, enter the opcodes that write to each structure (row) during the corresponding phase (column) of the instruction processing cycle. To make this simpler, let's only consider the following opcodes: AND, LDR, STR, and JMP.

Answer: Note that the book is a little unclear about when LDR and STR update the MAR, so we'll accept either "evaluate address" or "fetch operands."

| | FETCH | DECODE | EVAL ADDR | FETCH OPERANDS | EXECUTE | STORE |
|-----|-----------------|--------|-----------|----------------|---------|---------|
| PC | AND,LDR,STR,JMP | | | | JMP | |
| IR | AND,LDR,STR,JMP | | | | | |
| MAR | AND,LDR,STR,JMP | | LDR,STR | (LDR?,STR?) | | |
| MDR | AND,LDR,STR,JMP | | | LDR | | STR |
| RF | | | | | | AND,LDR |

6. [6 Points] **LC-3 Implementation.** Consider the logic diagram below showing part of the control structure of an LC-3 machine. N, Z, and P are condition codes.



- (a) What does the output of the NOR gate tell us?

Answer: The output of the NOR gate will be 1 if and only if the high-order four bits of the IR are 0 (*i.e.*, the IR contains a BR instruction).

- (b) What does the output of the OR gate tell us?

Answer: The output of the N, Z, or P AND gate will be 1 if and only both the condition code and the corresponding bit in the IR are 1. The output of the OR gate will be 1 if any condition code is set and the corresponding bit in the IR is also set.

- (c) What is the purpose of the signal labeled A?

Answer: The output A is 1 if and only if the instruction in the IR is a BR and the branch should take place because the appropriate condition code is set.