CS 250 Fall 2017 Homework 06 SOLUTION and GRADING GUIDE
Maximum score = 15 points

1. Text exercise 6.13. Your answer will have three parts as follows.
   a. Define the new BRR, Branch Relative, instruction by writing a line of C code. Use only labels shown in Figure 6.9 as variable names in your C statement.
   Answer: 32-bit_pgm._ctr. = 32-bit_pgm._ctr. + offset;

   b. Design the bit string for BRR and fill in the instruction format table here showing actual bit values where possible, "unused" where appropriate, and your chosen integer representation to be used in the Offset field.
   Answer: [3 points; 1 point for opcode, 1 point for all three register fields shown as unused (0 points if not all shown as unused), and 1 point for 2's comp offset]

	Opcode	Reg A	Reg B	Dst Reg	Offset
bits	31 – 27	26 – 23	22 – 19	18 – 15	14 – 0
BRR	Any unsigned integer other than 1, 2, 3, or 4	Unused	Unused	Unused	2's complement

   c. On Figure 6.9, provided below, overlay all the additional circuitry needed to execute BRR, according to your design (there is more than one way to add BRR to the hardware of Fig. 6.9). Use a color other than blue for your circuitry, so that your work clearly stands out.

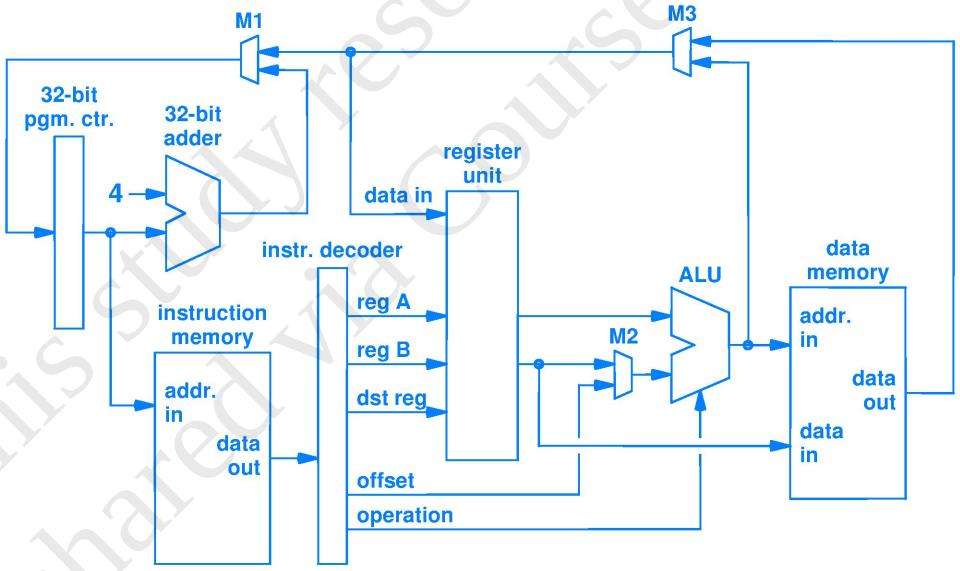

Figure 6.9 Illustration of data paths including data memory.

Answer: [3 points for either solution shown or for another correct solution. If you see another correct solution, forward it to me immediately. Thank you.]
One design solution is to use the ALU to compute PC+offset. This can be

accomplished by adding a new mux to select PC as an ALU input and then use the existing circuit in the same way as the JUMP instruction to complete BRR. The design circuit looks like this.

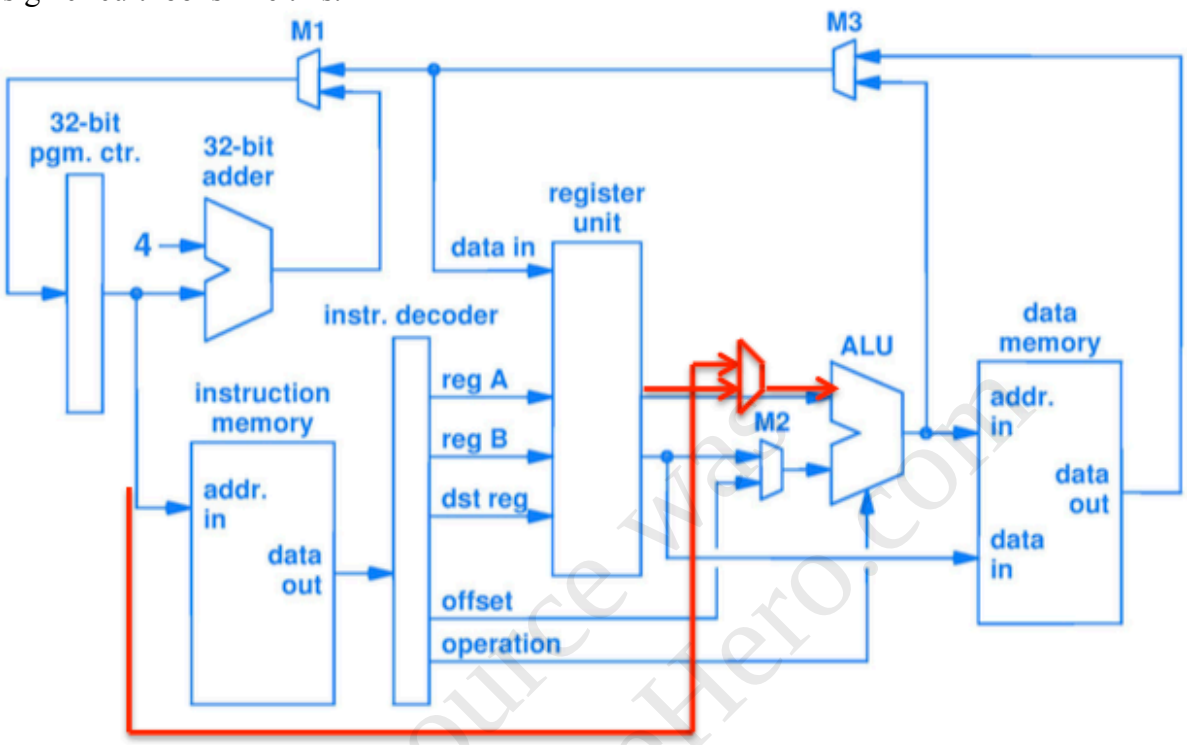

Figure 6.9 Illustration of data paths including data memory.

If you decide that a new adder should be included to compute PC+offset, then an alternative design is shown below. Mux M1 now has 3 inputs to select from, and needs a 2-bit control signal. This design is somewhat more expensive than the first one, increases the complexity of M1 control, but offers the advantage of reducing the propagation delay to obtain the actual next PC address by the delay for M3.

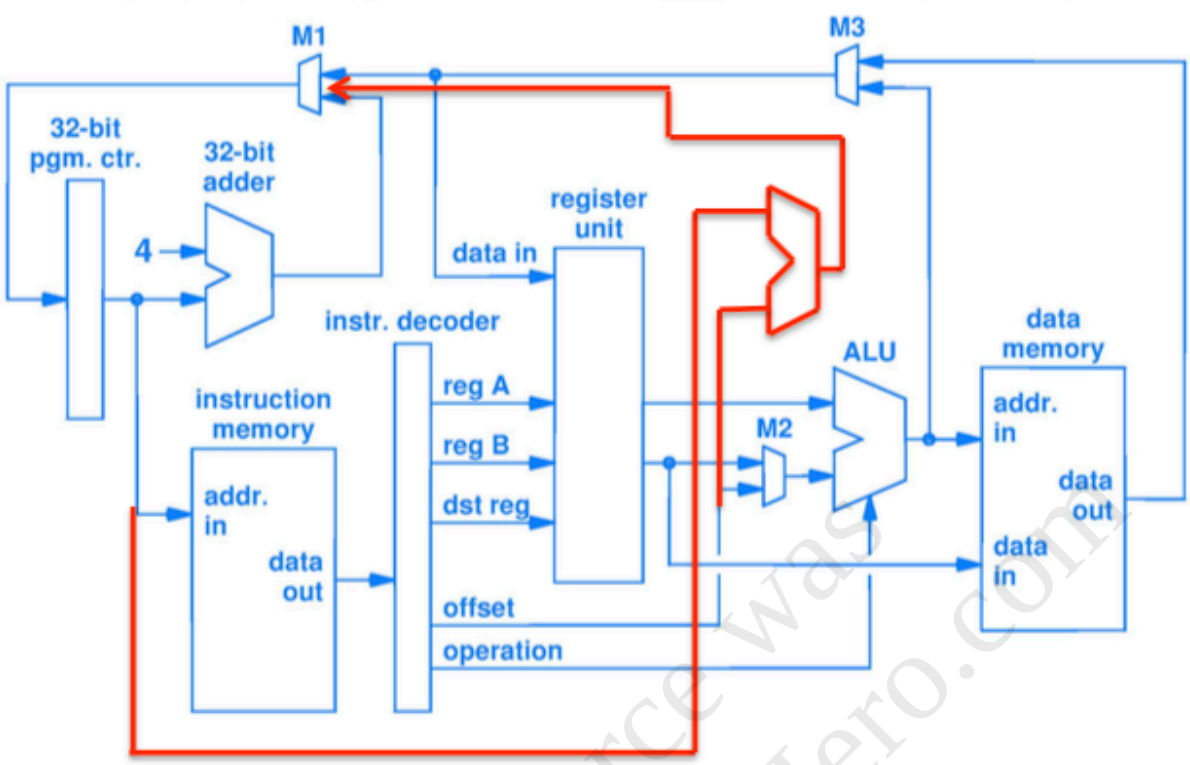

Figure 6.9 Illustration of data paths including data memory.

2. The BRR instruction (see Question 1 above) appearing in assembly as
   .LL2: BRR -36 ; branch to an address -36 (base 10) from the current address
   is part of a machine language instruction loaded into instruction memory. Assume .LL2 corresponds to 0x00400514; the opcode is 5; the computer has byte-addressed, Little Endian memory; and that the assembler used to generate the machine language fills unused fields with zeros.
   a. Fill in the table to show the contents of memory starting at .LL2.
      Answer: [3 points; 1 point for all addresses correct, 0.5 points for each correct stored bit string]

Memory address, 0x representation	Bit string stored, shown in binary representation
0x00400514	00100100
0x00400518	00000000
0x0040051C	00000000
0x00400520	00101000

   b. The computer performs a fetch at address 0x00400514. What bit string is produced at the output wires of the instruction memory? Put a space in your bit string after every fourth bit to help with readability.
      Answer: 0010 1000 0000 0000 0000 0000 0010 0100

c. If the instruction memory has 32 wires for output, and the bytes appear on those wires in Little Endian order, how is it that the instruction decoder can receive the machine language bit string fields in their proper arrangement as defined by the instruction format?
Answer: [3 points] The wires from instruction memory are permuted on their way to the instruction decoder such that the bit string arrives at the instruction decoder in the proper order.

3. The assembly language for a computer includes the JSR, Jump to SubRoutine, instruction defined as Current_instruction_pointer ← Current_instruction_pointer + Offset. A subroutine name in a high-level language program is represented in the assembly language code for that program as a label on the first assembly instruction of the subroutine. What replaces the subroutine name in machine language?
Answer: [3 points] At the machine language level, each call to the subroutine is performed by a JSR instruction. Because the first instruction of the subroutine will be stored at a fixed location in memory, while each different JSR instruction that calls this subroutine is stored at a different memory address, the Offset value is unique to each JSR instruction. Thus, a unique subroutine name in the high-level language has become a set of distinct immediate values, one for each JSR to the given subroutine, found in the Offset fields of all the JSR machine instructions.

4. Text exercise 9.2.
Answer: It would be typical to see a block comment before each routine and a comment on each line of code.

5. Text exercise 9.9. Also, state the size of the instructions in Figure 9.12.
Answer: The inserted jump instruction will jump (fetch at) the address of label2, which is 0x10. All instructions appearing in Figure 9.12 have a size of 4 bytes. If a new instruction is inserted before label1, then the addresses of all instructions following the inserted instruction must increase by 0x4.