

CS2214 COMPUTER ARCHITECTURE & ORGANIZATION FALL 2011

EXAM I ANSWERS

1) a) The table showing the values of registers and memory locations used by the code is as follows :

Instruction	PC	R8	R9	R10	R11	R12	M[1000 0000]	M[1000 2000]	M[1000 4000]	M[1000 6000]	Memory Accesses
Initial	40C000	?	?	?	10000000	1	6	8	C	?	-----
LW R8, 0(R11)	40C004	6	NS	NS	NS	NS	NS	NS	NS	NS	2 : IF & DR
LW R9, 2000(R11)	40C008	NS	8	NS	NS	NS	NS	NS	NS	NS	2 : IF & DR
LW R10, 4000(R11)	40C00C	NS	NS	C	NS	NS	NS	NS	NS	NS	2 : IF & DR
ADD R8, R8, R9	40C010	E	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
ADD R8, R8, R10	40C014	1A	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
SW R8, 6000(R11)	40C018	NS	NS	NS	NS	NS	NS	NS	NS	1A	2 : IF & DW
ADDI R11, R11, 4	40C01C	NS	NS	NS	10000004	NS	NS	NS	NS	NS	1 : IF
ADDI R12, R12, (-1) ₁₀	40C020	NS	NS	NS	NS	0	NS	NS	NS	NS	1 : IF
BNE R12, R0, (-9) ₁₀	40C024	NS	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF

The code adds respective elements of three vectors and then stores the result to another vector.

b) The new EMY instruction reads three memory locations, adds them and stores to another location. These locations are separated by 2000. We will call it ADD3M, meaning Add three memory locations :

Syntax : ADD3M Disp(Rs)

Semantics : $M[R_s + 3*Disp^+] \leftarrow M[R_s] + M[R_s + Disp^+] + M[R_s + 2*Disp^+]$

Format, etc. :

- **I** format is used since Disp is needed. Rt is **not** used.
- We make **five** memory accesses : One to fetch the instruction, three to read data and one to store data

The ADD3M instruction replaces **six** original instructions. The original code is rewritten with the ADD3M below :

40C000	AD3M	2000(R11)	# Add three locations. R11 initially has 10000000
40C004	ADDI	R11, R11, 4	# Increment the index register
40C008	ADDI	R12, R12, (-1) ₁₀	# Decrement the loop-end counter. R12 initially has 1
40C00C	BNE	R12, R0, (-4) ₁₀	# If not the end, go back to the ADD3M location

2) a) The table showing the values of registers and memory locations used by the code is below. The code checks if respective elements of two vectors are different, determines how many of them are different and stores the number in a location following the last element of the first vector.

b) The new EMY instruction compares a memory location with a register and if they are the same, it branches. We will call it BEQMR, meaning Branch if a memory location and a register are the same :

Syntax : BEQMR Rs, (Rt), Offset

Semantics : If $R_s = M[R_t]$ then $PC \leftarrow PC + (Offset^+ * 4)$

Format, etc. :

- **I** format is used since Offset is needed.
- We make **two** memory accesses :
 - One to fetch the instruction and one to read data

Instruction	PC	R8	R9	R10	R11	R12	M[1000 0000]	M[1000 0004]	M[1000 0008]	M[1000 5000]	M[1000 5004]	Mem. Acc.
Initial	400000	?	10000000	?	0	2	2A	F2	?	91	7C	-----
LW R8, 5000(R12)	400004	91	NS	NS	NS	NS	NS	NS	NS	NS	NS	2 : IF & DR
LW R10, 0(R9)	400008	NS	NS	2A	NS	NS	NS	NS	NS	NS	NS	2 : IF & DR
BEQ R8, R10, 1	40000C	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
ADDI R11, R11, 1	400010	NS	NS	NS	1	NS	NS	NS	NS	NS	NS	1 : IF
ADDI R9, R9, 4	400014	NS	10000004	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
ADDI R12, R12, (-1) ₁₀	400018	NS	NS	NS	NS	1	NS	NS	NS	NS	NS	1 : IF
BNE R13, R0, (-7) ₁₀	400000	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
LW R8, 5000(R12)	400004	7C	NS	NS	NS	NS	NS	NS	NS	NS	NS	2 : IF & DR
LW R10, 0(R9)	400008	NS	NS	F2	NS	NS	NS	NS	NS	NS	NS	2 : IF & DR
BEQ R8, R10, 1	40000C	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
ADDI R11, R11, 1	400010	NS	NS	NS	2	NS	NS	NS	NS	NS	NS	1 : IF
ADDI R9, R9, 4	400014	NS	10000008	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
ADDI R12, R12, (-1) ₁₀	400018	NS	NS	NS	NS	0	NS	NS	NS	NS	NS	1 : IF
BNE R13, R0, (-7) ₁₀	40001C	NS	NS	NS	NS	NS	NS	NS	NS	NS	NS	1 : IF
SW R11, 0(R11)	400020	NS	NS	NS	NS	NS	NS	NS	2	NS	NS	2 : IF & DW

The BEQMR instruction replaces **two** original instructions. The original code is rewritten with the BEQMR below :

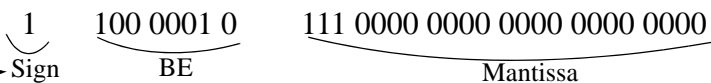
```

400000 LW      R8, 5000(R9)      # Load from the first vector. R9 has 10000000 initially
400004 BEQMR R8, (R9), 1      # If elements of first and second vectors are equal, skip one instruction
400008 ADDI   R11, R11, 1      # Increment the "Not equal" counter by 1. R11 is initially 0
40000C ADDI   R9, R9, 4        # Advance the pointer to the next location
400010 ADDI   R12, R12, (-1)10 # Decrement 1 from the loop-end counter. R12 is initially 2
400014 BNE   R12, R0, (-6)10 # If not the end, go back to location 400000
400018 SW    R11, 0(R9)        # Store the "not equal" counter value below the first vector

```

3) The decimal number corresponding to the IEEE-754 single-precision FP number is obtained as follows :

The number is a **negative** number, since the **sign** bit of the FP number is **one**.



BE → 1 0 0 0 0 1 0
7 6 5 4 3 2 1 0

$$BE = 2^7 + 2^1 = 130$$

$0 < BE < 255 \Rightarrow$ the number is normalized

$$e = BE - 127 = 130 - 127 = 3$$

Mantissa → 1 1 1 0000 0000 0000 0000 0000
-1 -2 -3-23

$$\text{Mantissa} = 2^{-1} + 2^{-2} + 2^{-3} = 1/2 + 1/4 + 1/8 = 0.5 + 0.25 + 0.125 = 0.875$$

$$(1 \ 1000010 \ 111000000000000000000000)_{\text{IEEE-754}} = (1.875 \times 2^3)_{10}$$