**Appendix F**

**Selected Solutions**

**F.5 Chapter 5 Solutions**

* 1. (a) ADD
     + operate
     + register addressing for destination and source 1
     + register or immediate addressing for source 2

1. JMP
   * + control
     + register addressing
2. LEA
   * + data movement
     + immediate addressing
3. NOT
   * + operate
     + register addressing

5.3 Sentinel. It is a special element which is not part of the set of allowable inputs and indicates the end of data.

5.5 (a) Addressing mode: mechanism for specifying where an operand is located.

1. An instruction’s operands are located as an immediate value, in a register, or in memory.
2. The 5 are: immediate, register, direct memory address, indirect memory address, base

+ offset address. An immediate operand is located in the instruction. A register operand is located in a register (R0 - R7). A direct memory address, indirect memory address and base + offset address all refer to operands locate in memory.

1. Add R2, R0, R1 => register addressing mode.

1

2 APPENDIX F. SELECTED SOLUTIONS

5.7 01111 (decimal 15)

5.9 (a) Add R1, R1, #0 => differs from a NOP in that it sets the CC’s.

1. BRnzp #1 => Unconditionally branches to one after the next address in the PC. There- fore no, this instruction is not the same as NOP.
2. Branch that is never taken. Yes same as NOP.

5.11 No. We cannot do it in a single instruction as the smallest representable integer with the 5 bits available for the immediate field in the ADD instruction is -16. However this could be done in two instructions.

5.13 (a) 0001 011 010 1 00000 (ADD R3, R2, #0 )

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| (b) | 1001 | 011 | 011 | 111111 | (NOT | R3, | R3 ) |
|  | 0001 | 011 | 011 | 1 00001 | (ADD | R3, | R3, #1 ) |
|  | 0001 | 001 | 010 | 0 00011 | (ADD | R1, | R2, R3 ) |

(c) 0001 001 001 1 00000 (ADD R1, R1, #0 )

or

0101 001 001 1 11111 (AND R1, R1, #-1)

1. Can’t happen. The condition where N=1, Z=1 and P=0 would require the contents of a register to be both negative and zero.

(e) 0101 010 010 1 00000 (AND R2, R2, #0)

5.15 1110 001 000100000 ( LEA R1, 0x20 ) R1 <- 0x3121

0010 010 000100000 ( LD R2, 0x20 ) R2 <- Mem[0x3122] = 0x4566

1010 011 000100001 ( LDI R3, 0x20 ) R3 <- Mem[Mem[0x3123]] = 0xabcd

0110 100 010 000001 ( LDR R4, R2, 0x1 ) R4 <- Mem[R2 + 0x1] = 0xabcd

1111 0000 0010 0101 ( TRAp 0x25 )

5.17 (a) LD: two, once to fetch the instruction, once to fetch the data.

1. LDI: three, once to fetch the instruction, once to fetch the data address, and once to fetch the data.
2. LEA: once, only to fetch the instruction.

5.19 PC-64 to PC+63. The PC value used here is the incremented PC value.

5.21 The Trap instruction provides 8 bits for a trap vector. That means there could be 28 = 256 trap routines.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 5.23 | x30ff | 1110 | 0010 0000 0001 | | | (LEA R1, #1) R1 <- 0x3101 |
|  | x3100 | 0110 | 010 001 00 0010 | | | (LDR R2, R1, #2) R2 <- 0x1482 |
| x3101 | | 1111 | 0000 | 0010 | 0101 | (TRAP 0x25) |
| x3102 | | 0001 | 0100 | 0100 | 0001 |  |
| x3103 | | 0001 | 0100 | 1000 | 0010 |  |

F.5. CHAPTER 5 SOLUTIONS 3

5.25 1001 100 011 111111 ;(NOT R4, R3)

0001 100 100 1 00001 ;(ADD R4, R4, #1)

0001 001 100 0 00 010 ;(ADD R1, R4, R2)

0000 010 000000101 ; (BRz Done)

0000 100 000000001 ; (BRn Reg3)

0000 001 000000010 ; (BRp Reg2)

0001 001 011 1 00000 ; (Reg3 ADD R1, R3, #0)

0000 111 000000001 ; (BRnzp Done)

0101 001 010 1 00000 ; (Reg2 ADD R1, R2, #0)

1111 0000 0010 0101 ; (Done TRAP 0x25)

5.27 Four different values: xAAAA, x30F4, x0000, x0005

5.29 (a) LDR R2, R1, #0 ;load R2 with contents of location pointed to by R1 STR R2, R0, #0 ;store those contents into location pointed to by R0

(b) The constituent micro-ops are: MAR < − SR

MDR < − Mem[MAR] MAR < − DR Mem[MAR] < − MDR

5.31 0x1000: 0001 101 000 1 11000

5.33 It can be inferred that R5 has exactly 5 of the lower 8 bits = 1.

5.35 The IR, SEXT unit, SR2MUX, Reg File and ALU implement the ADD instruction, alongwith NZP and the logic which goes with it.

5.37 Memory, MDR, MAR, IR, PC, Reg File, the SEXT unit connected to IR[8:0], ADDR2MUX, ADDR1MUX set to PC, alongwith the ADDER they connect to, and MAXMUX and GateMARMUX implement the LDI instruction, alongwith NZP and the logic which goes with it.

5.39 IR, PC, Reg File, the SEXT unit connected to IR[8:0], ADDR2MUX, ADDR1MUX set to PC, alongwith the ADDER they connect to, and MAXMUX and GateMARMUX implement the LEA instruction, alongwith NZP and the logic which goes with it.

5.41 (a) Y is the P Condition code.

(b) Yes. X should be one wherever the opcode field of the IR matches the opcodes which change the condition code registers. The problem is that X is 1 for the BR opcode (0000) and LEA (1110) in the given logic. They should be removed, and ADD opcode (0001) should be added.

5.43 Yes, there is difference. Instruction 1 (ADD) sets the Condition Codes while Instruction 2 (BR) doesn’t.

5.45 PC is put into MAR to fetch the next instruction that needs to be executed. PC is incremented to move to the next instruction to be fetched.

5.47

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **11** | **10** | **01** | **00** |
| **Mux 1 D[15:12]** | 3 | 2 | 1 | 0 |
| **Mux 2 D[11:8]** | 0 | 1 | 2 | 3 |
| **Mux 3 D[7:4]** | 3 | 1 | 2 | 0 |
| **Mux 4 D[3:0]** | 2 | 3 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| **R/W** | **MDR** | **MAR** |
| W | x72A3 | 00 |
| W | x8FAF | 11 |
| R | x72A3 | 00 |
| R | xFFFF | 10 |
| W | x732D | 11 |
| R | xFFFF | 01 |
| W | x37A3 | 01 |
| R | x37A3 | 01 |
| R | x732D | 11 |

Contents after accesses:

00: x72A3

01: x37A3

10: xFFFF

11: x732D

5.49 If R6 is zero, then R0 is used as a base register

5.51 State 13: State A: State B:  
 ALUK = NOT SR2SEL = REGISTER\_A; ALUK = ADD;

Gate\_ALU = 1 LD.REG = 1; LD.CC = 1

A close up of text on a white background

Description automatically generated 5.53

5.55 (Same answer as Exercise 6.27. Refer to Chapter 6 solutions)

5.57 1. ST, STR, STI  
 2. IR[5]  
 3. TRAP

5.59 ADD: 9  
AND: 9  
LD: 15  
LEA: 9  
LDI: 21  
NOT: 9  
BRnzp: 10  
TRAP: 15 + 21 = 36

5.61

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Inst # | Clock Cycle | Bus | State | Gate PC | Gate MDR | Gate ALU | Gate MARMUX | LD PC | LD MDR | LD MAR | LD CC | LD REG | DR MUX | MIO EN | R W | PC MUX |
| Inst 1 | T | x3010 | 18 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | - | - | - | PC+1 |
|  | T + 6 | xF0AB | 35 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 8 | x00AB | 15 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 9 | x1510 | 28 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | R7 | 1 | R | - |
|  | T + 10 | x1510 | 28 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | R7 | 1 | R | - |
|  | T + 11 | x1510 | 28 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | R7 | 1 | R | - |
|  | T + 12 | x1510 | 28 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 13 | x1510 | 28 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 14 | x1510 | 30 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | - | - | - | BUS |
| Inst 2 | T + 15 | x1510 | 18 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 21 | x2219 | 35 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 23 | x152A | 2 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 29 | x8001 | 27 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 11.9 | - | - | - |
| Inst 3 | T + 30 | x1511 | 18 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | T + 36 | x0804 | 35 | x | x | x | x | x | x | x | x | x | x | x | x | x |
| Inst 4 | T + 37 | x1516 | 18 | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | x | x1200 | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
|  | x | x0000 | x | x | x | x | x | x | x | x | x | x | x | x | x | x |