Solutions to Tutorial 2

2.1 Addressing Modes and Their Characteristics

(1) Solutions:

No.	Mnemonics	Destination	Source
1.	**`MOV R0,R1`**	Register Direct	Register Direct
2.	**`MOV R0,[0x100]`**	Register Direct	Absolute Addressing
3.	**`MOV R0,#0x100`**	Register Direct	Immediate Data
4.	**`MOV R1,#0x100`**	Register Direct	Immediate Data
5.	**`MOV R0,[R1]`**	Register Direct	Register Indirect
6.	**`MOV R0,[R2+7]`**	Register Direct	Register Indirect with Offset

(2) Solutions:

No.	Mnemonics	Instruction length (in words)	Cycle Count
1.	**`MOV R0,R1`**	**1**	**1** - 1 instruction read cycle (IRC)
2.	**`MOV R0,[0x100]`**	**2**	**3** - 2 IRC, 1 data read cycle (DRC)
3.	**`MOV R0,#0x100`**	**2**	**2** - 2 IRC
4.	**`MOV R1,#0x100`**	**2**	**2** - 2 IRC
5.	**`MOV R0,[R1]`**	**1**	**2** - 1 IRC, 1 DRC
6.	**`MOV R0,[R2+7]`**	**2**	**3** – 2 IRC, 1 DRC

(3) (a) **`R0`** = **`0x000`**, **`SR`** = **`0x002`**, **`PC`** = **`0x001`**.

(b) (i) Yes
(ii) absolute addressing and immediate data.
(iii) Yes for immediate data but not fro absolute addressing.

(c) (i) Yes
(ii) **`MOV R0,[R1]`**. Less instruction words to fetch during execution.
(iii) **`MOV R1,R0`**. Shortens the code and speeds up its execution

(d) (i) **`R0`** = **`0x005`** (machine code of the mnemonic **`MOV R0,[R1]`** at address 0x007)

2.2 Addressing Modes and Data Movement

Suggested solution:

No.	Mnemonics	Modified PC	Modified Register	Modified Memory	Comments
1.	MOV R0,R2	PC=0x001	R0=0x987 SR=0x004	Nil	N-flag set
2.	MOV R1,[R1]	PC=0x001	R1=0x000 SR=0x002	Nil	Z-flag set
3.	MOV #0x000,R1	Illegal			Immediate data always source operand only
4.	MOV [0x102],[0x103]	PC=0x003	SR=0x004	0x102= 0x800	N-flag set.
5.	MOV R0,[R3+0xFFE]	PC=0x002	R0=0xABC SR=0x004	Nil	Offset is -2
6.	MOV R1,0x100	Illegal			No such addressing mode.
7.	MOV SR,#0x001	PC=0x002	SR=0x001	Nil	

2.3 Arithmetic and Logical Instructions

(1) Suggested solution:

No.	Possible Mnemonics
1.	**MOV R0,#0x000**
2.	**MOVS R0,#0x000**
3.	**EOR R0,R0**
4.	**AND R0,#0x000**
5.	**SUB R0,R0**

(2) Suggested solution:

No.	Possible Mnemonics
1.	**ADD R1,#0x001**
2.	**INC R1**
3.	**MOVS R0,#0x001** **ADD R1,R0**

(3) **EOR R2,#0x111**.

(4)
```
MOV R1,R0
AND SR,0xFFE
RLC R0
ADD R0,R1
```

(i) **0x2AA** or decimal **682**.

(ii) 7A = (8A – A)

2.4 Assembly Program Analysis – Comparison and Counting

(1) **R0** = **0x007**, **R1** = **0x109** , **R2** = **0x003** , **R3** = **0xF80**, HotDays = **0x003**, **DaySum** = **0x007**

(2) The program computes the total number of days (**DaySum**) by counting number of word-sized values retrieved before the value -128 is encountered. It also counts the number of values that are larger than or equal to the decimal value 36 (i.e. **0x024**) and leave the result in memory variable **HotDays**.

(3) a) Use register **R2** to keep track of the number of hot days in order to **avoid unnecessary memory access**.

b) Use the more **optimized** **inc R0** and **inc R1** instructions.

c) We could use **EOR R0,R0** or **MOVS R0,#0** instead of **MOV R0,#0**.

(4) Replace **JHS** instruction with the **JGE** instruction.