

Faculty of Engineering and Technology

Electrical and Computer Engineering Department

Computer Organization and Assembly Language

**Assignment No.3 & Assignment No.4**

Chapter 12 & Chapter 13

**Prepared by:**

|  |  |  |  |
| --- | --- | --- | --- |
| Student Name: | Nour Alaydi | Student ID: | 1222810 |
| Student Name: | Yuna Nawahda | Student ID: | 1211524 |

**Deadline: 21/1/2024**

|  |
| --- |
|  |

**Rules and Regulations:**

You are required to:

1. fill name(s), ID(s) and date in this cover page.
2. answer clearly all problems in the assignment.
3. write the name/names of the participating student(s) besides each problem. The participation percentage should be equal between you and your partner (if you have).
4. avoid adding your answers as snapshots; the credits of the answers will be zero.
5. avoid cheating from other students, solution manual, and internet.
6. submit .pdf file (i.e. ID1\_ID2.pdf) for the solution by accessing your RITAJ and replying on the assignment before the deadline, late submissions are not allowed.
7. do the submission by you or your partner, one submission per group is enough. If you submit twice or more, the last submission will be considered only.

**Assignment No.3 - Chapter 12 (***10th edition***)**

**Review Questions:**

**Question#1 (Q12.1 P446)**

What are the typical elements of a machine instruction?

**Answer#1 (*NAME1, NAME2*):**

* Operation code (opcode)
* Source operand reference
* Result operand reference
* Next instruction reference

**Question#2 (Q12.2 P446)**

What types of locations can hold source and destination operands?

**Answer#2 (*NAME1, NAME2*):**

* Main or virtual memory
* Processor register
* Immediate
* I /O device

**Question#3 (Q12.11 P446)**

List three possible places for storing the return address for a procedure return.

**Answer#3 (*NAME1, NAME2*):**

• registers

• start of called procedure

• top of stack

**Question#4 (Q12.14 P446)**

What is the difference between big endian and little endian? Support your answer with examples for the two types.

**Answer#4 (*NAME1, NAME2*):**

**Big endian** stores the most significant bytes first, whereas **little endian** stores the least significant bytes first.

**For example**, the 32-bit hexadecimal number **0x12345678**

in the big endian: **|12 | 34 | 56 | 78**

in the little endian**: | 78 | 56 | 34 | 12 |**

**Chapter 12 (***10th edition***):**

**Problems:**

**Problem#1 (P12.3 P446-447)**

A given microprocessor has words of 1 byte. What is the smallest and largest integer that can be represented in the following representations?

a. Unsigned

b. Sign-magnitude.

c. Unsigned packed decimal.

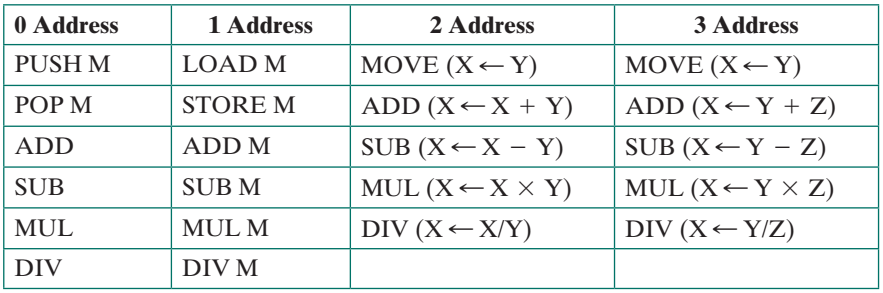
d. Signed packed decimal.

**Answer#1 (*NAME1, NAME2*):**

1. Unsigned: 0, 255
2. Sign-magnitude: -127, 127
3. Unsigned packed decimal: 0 ,99
4. Signed packed decimal: -9 ,99

**Problem#2 (P12.3 P446-447)**

Compare zero-, one-, two-, and three-address machines by writing programs to compute X = (A + B \* C)/(D - E \* F) for each of the four machines. The instructions available for use are as follows:



**Answer#1 (*NAME1, NAME2*):**

|  |  |  |  |
| --- | --- | --- | --- |
| 0 Adress | 1 Address | 2 Address | 3 Address |
| PUSH A  PUSH B  PUSH C  MUL  ADD  PUSH D  PUSH E  PUSH F  MUL  SUB  DIV  POP X | LOAD A  ADD B  MUL C  LOAD D  SUB E  MUL F  DIV TMP  STORE X | MOVE R1, B  MUL R1, C  ADD R1, A  MOVE R2, E  MUL R2, F  MOVE R3, D  SUB R3, R2  DIV R1, R3 | MUL R1, B, C  ADD R2, R1, A  MUL R3, E, F  SUB R4, D, R3  DIV R5, R2, R4 |

**Assignment No.4 - Chapter 13 (***10th edition***):**

**Review Questions:**

**Question#1 (Q13.8 P484)**

What is the advantage of autoindexing?

**Answer#1 (*NAME1, NAME2*):**

**Advantages of Autoindexing:**

• Auto indexing act as key to information retrieval.

• Register reference in indexing is sometimes implicit and sometimes explicit.

• Provides efficient mechanism for performing iterative operations.

• With Autoindexing, document in websites can be searched easily and at faster rate.

• Automated indexing

- easily retrievable,

-Modified and

-Revisited when any errors are noticed.

• Information resources are shared by information centers using automated indexing.

• Auto indexing supports exact matches when searching any document in websites.

• Indexing can be performed either before indirection or after indirection which is known as preindexing and postindexing.

**Question#2 (Q13.9 P484)**

What is the difference between postindexing and preindexing? Provide your answer with examples.

**Answer#2 (*NAME1, NAME2*):**

|  |  |
| --- | --- |
| **Preindexing** | **Postindexing** |
| Preindexing involves direct addressing and indexing | Postindexing also uses direct addressing and indexing |
| Indexing is performed before indirection | Indexing is performed after indirection |
| Base register contains memory address | Memory address is base register value. |
| Base register value can be updated by incrementing or decrementing the offset value (i.e. updated base register). | Offset can be added or subtracted using base register value (i.e. updated base register). |
| Sum of base address plus offset yields effective address. | Base address stored in base register is used directly as effective address |
| Result is stored in updated base register | Result is stored in original base register |

**Question#3 (Q13.11 P484)**

What are the advantages and disadvantages of using a variable-length instruction format?

**Answer#3 (*NAME1, NAME2*):**

**Advantages**: It easy to provide a large repertoire of opcodes, with different opcode lengths. Addressing can be more flexible, with various combinations of register and memory references plus addressing modes.

**Disadvantages**: an increase in the complexity of the CPU.

**Chapter 13 (***10th edition***):**

**Problems:**

**Problem#1 (P13.2 P485)**

Let the address stored in the program counter be designated by the symbol X1. The instruction stored in X1 has an address part (operand reference) X2. The operand needed to execute the instruction is stored in the memory word with address X3. An index register contains the value X4. What is the relationship between these various quantities if the addressing mode of the instruction is (a) direct; (b) indirect; (c) PC relative; (d) indexed?

**Answer#1 (*NAME1, NAME2*):**

**a.** Direct mode: X3 = X2

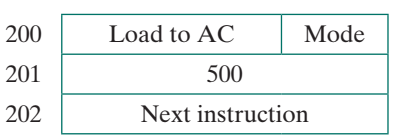
**b**. Indirect mode: X3 = (X2)

**c.** PC relative: X3 = X1 + X2 + 1

**d**. Indexed: X3 = X2 + X4

**Problem#2 (P13.4 P485)**

Consider a 16-bit processor in which the following appears in main memory, starting at location 200:



The first part of the first word indicates that this instruction loads a value into an accumulator. The Mode field specifies an addressing mode and, if appropriate, indicates a source register; assume that when used, the source register is R1, which has a value of 400. There is also a base register that contains the value 100. The value of 500 in location 201 may be part of the address calculation. Assume that location 399 contains the value 999, location 400 contains the value 1000, and so on. Determine the effective address and the operand to be loaded for the following address modes:

**a**. Direct **b**. Immediate **c**. Indirect **d**. PC relative **e**. Displacement **f**. Register **g**. Register indirect **h**. Autoindexing with increment, using R1

**Answer#2 (*NAME1, NAME2*):**

1. **Direct:**

 Effective address is available at instruction itself.  
Ac = 201 🡪 500  
**The Effective Address = 500**  
**Operand at 500 = 1100**

1. **Immediate**:

This mode has an operand field rather than the address field.  
Ac = 201  
Value = 500  
**The Effective Address = 201**  
**Operand at 201 = 500**

1. **Indirect**:

In Indirect mode, the address field of instruction provides the address of the effective address which is stored in memory.

Ac = 201  
**Effective Address = ADD[ADD[201]]  
 = ADD[500]  
 = 1100**  
**Operand at 1100 = 1700**

**d) PC relative:**In this Program Counter is added to address part of the instruction to obtain the effective address.

Accumulator = 201  
PC = 201 + 1 (address for next instruction)  
**Effective Address = PC + Value at 201**  
**= 202 + 500**  
**= 702**  
**Operand at 702 = 1302**

**e) Displacement Addressing mode:**

in this Index register/ base register is added to the address part of instruction.

**Effective Address = Base Address + ADD[AC]**

**= 100 + ADD[201]  
= 100 + 500  
= 600  
Operand at 600 = 1200**

**f) Register:**

Instruction consists of Opcode and Address of register  
**Effective Address = R1**  
**Operand at R1 = 400**

**g) Register indirect:**

In this, Address of Register holds the Address of Operand which is in the memory.  
**Effective address = ADD[R1] = 400**  
**Operand at 400 = 1000**

**h) Autoindexing with increment, using R1:**

In this, the register R1 is incremented after its value is used.  
**Effective address = ADD[R1]   
= 400  
Operand at 400 = 1000**  
**Value at R1 = 401**

☺ Good luck ☺

Ibrahim A. Nemer