**Talal Jawaid**

**2/19/19**

**CSC 137**

**Professor Chang**

**TTH 5:30-6:45 P.M**

**5.1**

**A computer uses a memory unit with 256K words of 32 bits each. A binary**

**instruction code is stored in one word of memory. The instruction has four**

**parts: an indirect bit, an operation code, a register code part to specify one**

**of 64 registers, and an address part.**

**a. How many bits are there in the operation code, the register code part,**

**and the address part?**

Address : 18 bits

Register Code: 6 bits

Opcode: 7 bits

**5.2**

**What is the difference between a direct and an indirect address instruction?**

Direct address instructions requires two references to memory.

These references are the read instruction and the read operand

Indirect address instruction requires one more reference, the read effective address.

**5.3**

**The following control inputs are active in the bus system shown in Fig. 5-4. For each case, specify the register transfer that will be executed during the next clock transition.**

1. The memory is read to the bus and then loaded to IR. IR is fed in from M[AR]
2. TR is sent to bus and load to PC. TR is into PC
3. AC to bus, then written to memory, load to DR

AC into DR, AC into M[AR]

1. ADD DR to AC: AC <- AC + DR

**5.4**

**The following register transfers are to be executed in the system of Fig. 5-4. For each transfer, specify: (1) the binary value that must be applied to bus select inputs S2f Si, and S0; (2) the register whose LD control input must be active (if any); (3) a memory read or write operation (if needed); and (4) the operation in the adder and logic circuit (if any).**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | S2,S1,S0 | LD | MEM | ADDER |
| (a) | AR <- PC | 010 – PC | AR |  |  |
| (b) | IR <- M[AR] | 111 – M | IR | R |  |
| © | M[AR] <- TR | 110 – TR |  | W |  |
| (d) | DR<- AC  AC <- DR | 100 – AC | DR and AC |  | Transfer DR to AC |

**5.5**

**Explain why each of the following microoperations cannot be executed 168 CHAPTER FIVE Basic Computer Organization and Design during a single clock pulse in the system shown in Fig. 5-4. Specify a sequence of microoperations that will perform the operation**

1. **IR <- M[PC]** Pc can’t provide addr to mem. Addr has to be transferred to AR before

**AR<-PC**

**IR<-M[AR]**

1. **AC <- AC + TR**

Add op must be done with DR. Transfer TR to DR before

DR <- TR

AC <- AC + DR

**(c)** DR <- DR + AC (AC doesn’t change) add operations should be stored in AC not DR.

**5.6**

**a)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0001 | 0000 | 0010 | 0100 |  |
| Hex = | 1 | 0 | 2 | 4 |

ADD M[024] to AC

**b)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1011 | 0001 | 0010 | 0100 |  |
| Hex = | B | 1 | 2 | 4 |

Stores AC in M[M[124]]

**c)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | 0111 | 0000 | 0010 | 0000 | |  | | |
| Hex = 7 0 2 0 |  |  |  | |  |

Increments AC

**5.7**

The two instructions needed are the CLE Clear E and

CME Complement E instructions

**5.9**

**AC = Accumulator; IR = Instruction Register;**

**AR = Address Register;  PC = Program Counter;**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **E** | **AC** | **PC** | **AR** | **IR** |
| **Initial** | 1 | A937 | 021 |  |  |
| **CLA** | 1 | 0000 | 022 | 800 | 7800 |
| **CLE** | 0 | A937 | 022 | 400 | 7400 |
| **CMA** | 1 | 56C8 | 022 | 200 | 7200 |
| **CME** | 0 | A937 | 021 | 100 | 7100 |
| **CIR** | 1 | D49B | 022 | 080 | 7080 |
| **CIL** | 1 | 526F | 022 | 040 | 7040 |
| **INC** | 1 | A938 | 022 | 020 | 7020 |
| **SPA** | 1 | A937 | 022 | 010 | 7010 |
| **SNA** | 1 | A937 | 023 | 008 | 7008 |
| **SZA** | 1 | A937 | 022 | 004 | 7004 |
| **SZE** | 1 | A937 | 022 | 002 | 7002 |
| **HLT** | 1 | A937 | 022 | 001 | 7001 |

**5.10**

**AC = Accumulator; AR = Address Register; PC = Program Counter;**

**DR = Data Register; IR = Instruction Register;**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **PC** | **AR** | **DR** | **AC** | **IR** |
| **Initial** | 021 |  |  | A937 |  |
| **AND** | 022 | 083 | B8F2 | A832 | 0083 |
| **ADD** | 022 | 083 | B8F2 | 6229 | 1083 |
| **LDA** | 022 | 083 | B8F2 | B8F2 | 2083 |
| **STA** | 022 | 083 |  | A937 | 3083 |
| **BUN** | 083 | 083 |  | A937 | 4083 |
| **BSA** | 084 | 084 | = | A937 | 5083 |
| **ISZ** | 022 | 083 | B8F3 | A937 | 6083 |

**5.11**

**AR = Address Register; SC = Sequence Counter; PC = Program Counter;**

**DR = Data Register; IR = Instruction Register;**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **PC** | **AR** | **DR** | **AC** | **SC** |
| **Initial** | 7FF |  |  |  | 0 |
| **T0** | 7FF | 7FF |  |  | 1 |
| **T1** | 800 | 7FF |  | EA9F | 2 |
| **T2** | 800 | A9F |  | EA9F | 3 |
| **T3** | 800 | C35 |  | EA9F | 4 |
| **T4** | 800 | C35 | FFFF | EA9F | 5 |
| **T5** | 800 | C35 | 0000 | EA9F | 6 |
| **T6** | 801 | C35 | 0000 | EA9F | 0 |

**5.12**

**a).** 3AF | 932E         9 = 1001 I = 1 001 = ADD, so next is to ADD 32E

    32E | 09AC

    9AC | 8B9F

**b).** 32E | 09AC        AC = 7EC3   7EC3 + 8B9F = 0A62

    9AC | 8B9F         DR = 8B9F