| | | | |
|---|---|---|---|
| | ***IGEE***<br>***University M'Hamed***<br>***Bougara of Boumerdès*** | ***Last Name:*** ……<br>***First Name:*** ……<br>***Group:*** …… | ***Grade :*** |

***EE321: Computer Architecture***
***Final Exam (1h30)***
***2015 – 2016***

**Notes: Answer briefly and clearly using the provided space. No extra sheet will be accepted.**

## 1. Memory (7 pts)

Consider the memory block diagram given in Figure 1. We assume one square cell array.

1.1 How many bit-lines and word-lines do we have? Justify.

1.2 How many address bits are required for the row decoder? Justify.

1.3 How many address bits are required for the column decoder? Justify.

1.4 If we assume the memory is designed using 8 cell arrays (2 rows by 4 colums), give the number of bit-lines and word-lines for each cell array. Justify.

1.5 Assuming 8 cell arrays, for each cell array, how many address bits are required for: a) row decoder, and b) column decoder? Justify.

1.6 Give the steps, and for each step, the values of the input signals (address, data, and control signals) to read the content of location 0F8AH.

1.7 Assume that a 128Kx32-bit main memory is built using this memory chip, find the number of the used memory chips to build the main memory. Justify.

## 2. Control Unit (6 pts)

Consider the simple 12-bit CPU in Figure 2 designed using a microprogrammed control unit.

2.1 Why microprogrammed control unit is easy to update than hardwired control unit?

2.2 What is the advantage and disadvantage of horizontal microinstruction compared to vertical microinstruction?

2.3 Which fields do we have in horizontal microinstruction?

2.4 Based on the simple CPU architecture, how many bits do we need for the internal CPU control signals field? Justify.

2.5 If we assume our control unit has 240 horizontal microinstructions, hown many bits do we need for the next microinstruction address field?

2.6 For next microinstruction address computation, which signals are used for condition bits?

## 3. Program Execution (7 pts)

Consider the simple 12-bit CPU given in Figure 2. The CPU has 12-bit address bus and supports the opcodes given in Table 2. The micro-operations of the fetch cycle, interrupt cycle, and execute cycle of the ADR instruction are given in Table 1. The binary codes of the general purpose registers (GPR) are given in Table 4, and the instruction format of the data processing operation is given in Table 5.

3.1 How many instructions and how many GPR can this processor support? Justify.

3.2 Give the hex code of the instruction: a)**HLT**, b) **ADR AC,R0**, and c) **LDP R0,0FH**. Justify.

3.3 How many memory locations we can address using instruction: a) **LDI Rd,X**, b) **LDD Rd,X**, c) **LDP Rd,X**, and d)**MV Rd,Rs**? Justify.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

3.4 For the **MV Rd,Rs** instruction, why: a) Rd can be any GPR except AC, and b) Rs can be any GPR except IR and MAR?

.....
.....
.....
.....
.....
.....
.....

3.5 Consider the memory content in Table 3. Fill the following table with the contents of the registers after the execution of the instruction stored at address 803H. Put X if unknown.

| *Register* | *AC* | *PC* | *MBR* | *MAR* | *IR* | *R0* | *R1* | *R2* | *R3* |
|---|---|---|---|---|---|---|---|---|---|
| *Before* | *001H* | | | | | *80AH* | *809H* | *808H* | *807H* |
| *After* | | | | | | | | | |

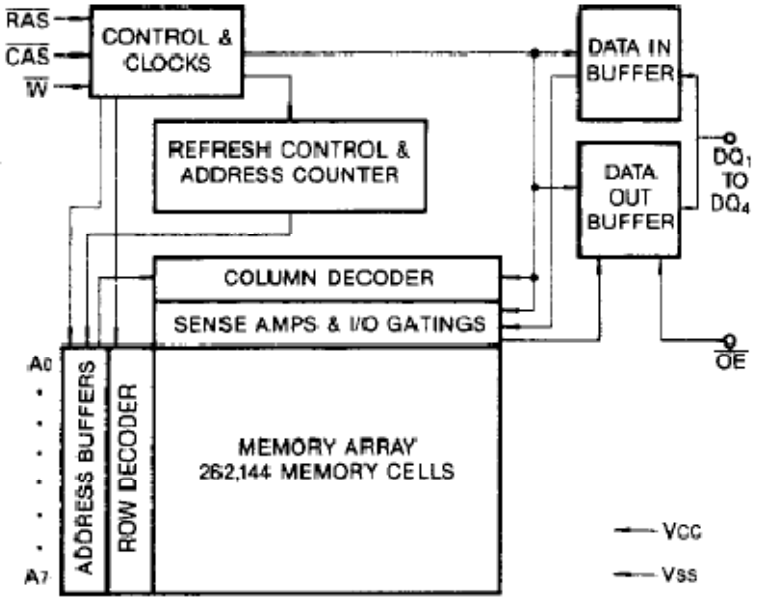
3.6 Give the micro-operations and time units for the <u>execute</u> cycle of the instruction stored at location 803H.

.....
.....
.....
.....
.....
.....
.....

3.7 Fill the following table with the contents of the registers at each time unit of the <u>execute</u> cycle of the instruction stored at address 803H.

| *Register* | *AC* | *PC* | *MBR* | *MAR* | *IR* | *R0* | *R1* | *R2* | *R3* |
|---|---|---|---|---|---|---|---|---|---|
| *End fetch* | *001H* | | | | | *80AH* | *809H* | *808H* | *807H* |
| *T1* | | | | | | | | | |
| *T2* | | | | | | | | | |
| *T3* | | | | | | | | | |
| *T4* | | | | | | | | | |

***Good Luck!***




| Pin Name | Pin Function |
|---|---|
| $A_0$-$A_7$ | Address Inputs |
| $\overline{RAS}$ | Row Address Strobe |
| $\overline{CAS}$ | Column Address Strobe |
| $\overline{W}$ | Read/Write Input |
| $\overline{OE}$ | Output Enable |
| $DQ_1$-$DQ_4$ | Data In/Out |
| $V_{CC}$ | Power (+5V) |
| $V_{SS}$ | Ground |

**Figure 1: Memory block diagram [© Samsung].**

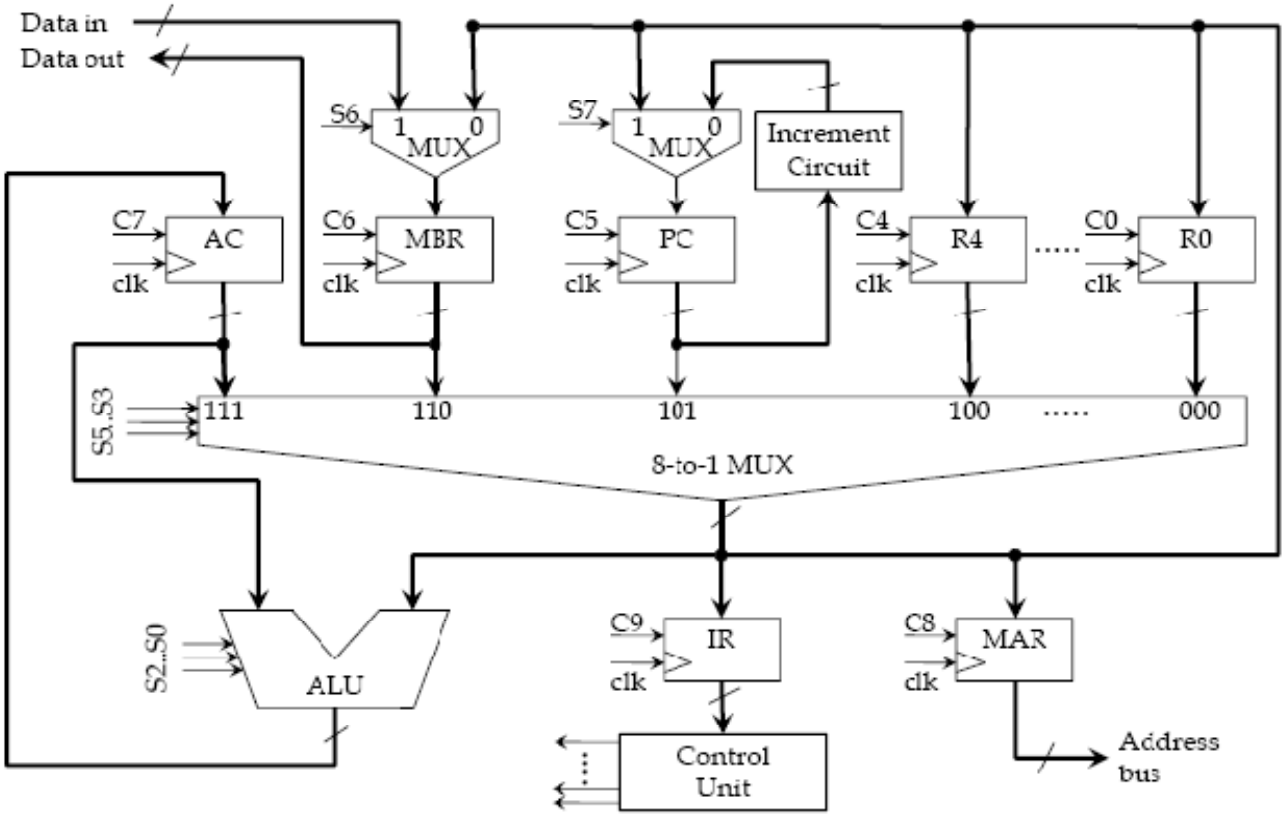



**Figure 2: Simple CPU architecture.**

| Clock cycle | Fetch cycle | Interrupt cycle | Execute cycle of ADR instruction (i.e. ADR AC, Rs) |
|---|---|---|---|
| T1 | *MAR ← PC* | *MBR ← PC* | *AC ← AC + Rs* |
| T2 | *MBR ← data in* | *MAR ← save address* | |
| T3 | *IR ← MBR*<br>*PC ← PC+1* | *Memory ← MBR*<br>*PC ← ISR address* | |

**Table 1: Micro-opérations of fetch cycle, interrupt cycle, and execute cycle of the ADR instruction.**

| Instruction | Opcode | Description |
|---|---|---|
| LDI Rd, X | 1H | Rd = X |
| LDD Rd, X | 2H | Rd = [X] |
| LDP Rd, X | 3H | Rd = [R0+X] |
| LDR Rd, Rs | 4H | Rd = [Rs] |
| STR Rd, Rs | 5H | [Rd] = Rs |
| ADR AC, Rs | 6H | AC = AC+Rs |
| ADM AC,X | 7H | AC = AC+X |
| ADD AC, X | 8H | AC =AC+ [X] |
| ADN AC,X | 9H | AC=AC+[[X]] |
| ADP AC,X | AH | AC=AC+[R0+X] |
| MV Rd, Rs | BH | Rd = Rs |
| HLT | FH | Stop the execution of the program |

*Rd = Destination register (it can be any GPR except AC).*
*Rs = Source register (it can be any GPR except IR and MAR)*
*R0 = Register R0.*
*AC = Accumulator.*
*[X] = content of memory location X.*

**Table 2: Instructions description.**

| Address (hex) | Content (hex) |
|---|---|
| 800H | 10FH |
| 801H | 21EH |
| 802H | 32AH |
| 803H | A73H |
| 804H | B37H |
| 805H | 543H |
| 806H | F05H |
| 807H | F06H |
| 809H | F07H |
| 80AH | F08H |
| 80BH | F09H |
| 80CH | F10H |
| 80DH | F11H |
| 80EH | F02H |
| 80FH | F03H |

**Table 3: Memory Content.**

| Register | R0 | R1 | R2 | R3 | R4 | PC | MBR | AC | IR | MAR |
|---|---|---|---|---|---|---|---|---|---|---|
| Code | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 |

**Table 4: Binary codes of GPR.**

| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Opcode | | | | Destination register | | | | Source operand | | | |

*Source operand can be register code or value.*
*Destination register is the code of the register used for the operation result.*

**Table 5: Data processing instruction format.**