University of Patras

DEPARTMENT OF COMPUTER & INFORMATION ENGINEERING

COMPUTER ARCHITECTURE LABORATORY

EXERCISE 2nd

The registers given in the workshop are as follows:

• Program Counter (PC): 10002

• Accumulator (ACC): 10012

• Auxiliary register X: 01112

**MACRO COMMAND ANALYSIS AND MICROCOMMAND DESCRIPTION**

**Bootstrap: First we define 2 bootstrap commands, m00 and m01, which initialize the PC with the address of the main memory where we saved the first command of the macro program.**

**m00: SW + 0 → PC, MAR**

**m01: NEXT(PC)**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m00 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1000 | xx | x111010111 |
| m01 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**LDA #K: Load the hexadecimal number K into the Accumulator.**

**PC + 1 → PC, MAR**

**It increments the contents of PC by one and addresses main memory to carry the integer, which is the hexadecimal number K.**

**MDR + 0 → ACC**

**Stores the contents of the MDR, i.e. the hexadecimal number K, in the Accumulator.**

**PC + 1 → PC, MAR**

**It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.**

**NEXT (PC)**

**µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m02 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m03 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1001 | xx | x110011101 |
| m04 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m05 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**LDΧ #K: Load the X auxiliary register with the hexadecimal number K.**

**PC + 1 → PC, MAR**

**It increments the contents of PC by one and addresses main memory to carry the integer, which is the hexadecimal number K.**

**MDR + 0 → X**

**Stores the contents of MDR, i.e. the hexadecimal number K, in the auxiliary register X.**

**PC + 1→ PC, MAR**

It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.

1. NEXT(PC)

µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m06 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m07 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 0111 | xx | x110011101 |
| m08 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m09 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**LDA ($K): Load into the Accumulator the contents of the memory location at address the contents of the memory location at address K.**

**PC + 1 → PC, MAR**

**It increments the contents of PC by one and addresses main memory to carry the integer, i.e. the hexadecimal number K.**

**MDR + 0 → NOP, MAR**

**It addresses the main memory with the contents of the MDR, i.e. the K number.**

**MDR + 0 → NOP, MAR**

**Addresses main memory with the contents of MDR, i.e. the contents of the memory location with address K.**

**MDR + 0 → ACC**

**The contents of MDR, i.e. the contents of the memory location with address the contents of the memory location with address K, are stored in the ACC.**

**PC + 1 → PC, MAR**

**It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.**

**NEXT (PC)**

**µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m0a | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m0b | xxxxx | 000 | xxx | 111 | 000 | 001 | xxxx | xxxx | xx | xx11011101 |
| m0c | xxxxx | 000 | xxx | 111 | 000 | 001 | xxxx | xxxx | xx | xx11011101 |
| m0d | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1001 | xx | x110011101 |
| m0e | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m0f | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

LDAX: Load into the Accumulator the contents of the memory location addressed by the contents of the X register.

X + 0 → NOP, MARAddresses main memory with the contents of the X register.

MDR + 0 → ACC

The contents of the MDR, that is, the contents of the memory location addressed by the contents of the X register, are stored in the Accumulator.

PC + 1 → PC, MAR

It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.

NEXT (PC)

µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m10 | xxxxx | 000 | xxx | 100 | 000 | 001 | 0111 | xxxx | xx | xx1101x1xx |
| m11 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1001 | xx | x110011101 |
| m12 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m13 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**LDA $K, X: ​​Load into the Accumulator the contents of the memory location with address equal to R(X)+K (sum of the hexadecimal number K and the contents of the auxiliary register X).**

**PC + 1 → PC, MAR**

**It increments the contents of PC by one and addresses main memory to carry the integer, i.e. the hexadecimal number K.**

**MDR + X → NOR, MAR**

**The sum of MDR, i.e. the hexadecimal number K, and the contents of auxiliary register X is transferred to MAR and addresses main memory.**

**MDR + 0 → ACC**

**Stores in the Accumulator the contents of the memory location with address equal to R(X)+K.**

**PC + 1 → PC, MAR**

It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.

NEXT (PC)

µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m14 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m15 | xxxxx | 000 | xxx | 101 | 000 | 001 | 0111 | xxxx | xx | xx11011101 |
| m16 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1001 | xx | x110011101 |
| m17 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m18 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

LDA ($K, X): Load into the Accumulator the contents of the memory location with address equal to M(K)+R(X) (sum of the contents of the memory location with address K and the contents of the auxiliary register X).

PC + 1 → PC, MAR

It increments the contents of PC by one and addresses main memory to carry the integer, i.e. the hexadecimal number K.

MDR + 0 → NOP, MAR

It addresses the main memory with the contents of the MDR, i.e. the K number.

MDR + X → NOP, MAR

The sum of MDR, i.e. the contents of address K, and the contents of auxiliary register X is transferred to MAR and addresses main memory.

MDR + 0 → ACC

Stores in the Accumulator the contents of the memory location with address equal to M(K)+R(X).

PC + 1 → PC, MAR

It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.

NEXT (PC)

µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m19 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m1a | xxxxx | 000 | xxx | 111 | 000 | 001 | xxxx | xxxx | xx | xx11011101 |
| m1b | xxxxx | 000 | xxx | 101 | 000 | 001 | 0111 | xxxx | xx | xx11011101 |
| m1c | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1001 | xx | x110011101 |
| m1d | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m1e | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

STA $K: Store the contents of the Accumulator in the memory location with address K.

PC + 1 → PC, MAR

It increments the contents of PC by one and addresses main memory to carry the integer, i.e. the hexadecimal number K.

MDR + 0 → NOP, MAR

Addresses main memory with the contents of MDR, i.e. the hexadecimal number K.

ACC + 0 → NOP, MWE~

The contents of the Accumulator are stored at the main memory address pointed to by MAR, i.e. address K.

PC + 1 → PC, MAR

It increments the contents of PC by one and addresses main memory to point to the opcode of the next macro and continue with its execution.

NEXT (PC)

µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | BRA  (4:0) | BIN  (2:0) | CON (2:0) | I (2:0) | I (5:3) | I (8:6) | APORT (3:0) | BPORT (3:0) | DDATA (2:0) | Control Signals |
| m1f | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m20 | xxxxx | 000 | xxx | 111 | 000 | 001 | xxxx | xxxx | xx | xx11011101 |
| m21 | xxxxx | 000 | xxx | 100 | 000 | 001 | 1001 | xxxx | xx | xx0001x1xx |
| m22 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1000 | 1000 | 01 | x111011110 |
| m23 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

MICROCODE

The contents of micro memory, mapper and main memory are as follows:

MICRO

m00 00000 000 000 111 000 011 0000 0001 00 0111010111 BOOTSTRAP

m01 00000 000 000 000 000 001 0000 0000 00 0110001111

m02 00000 000 000 101 000 011 0001 0001 01 0111011110 LDA #K

m03 00000 000 000 111 000 011 0000 0000 00 0110011101

m04 00000 000 000 101 000 011 0001 0001 01 0111011110

m05 00000 000 000 000 000 001 0000 0000 00 0110001111

m06 00000 000 000 101 000 011 0001 0001 01 0111011110 LDX #K

m07 00000 000 000 111 000 011 0000 0010 00 0110011101

m08 00000 000 000 101 000 011 0001 0001 01 0111011110

m09 00000 000 000 000 000 001 0000 0000 00 0110001111

m0a 00000 000 000 101 000 011 0001 0001 01 0111011110 LDA ($K)

m0b 00000 000 000 111 000 001 0000 0000 00 0011011101

m0c 00000 000 000 111 000 001 0000 0000 00 0011011101

m0d 00000 000 000 111 000 011 0000 0000 00 0110011101

m0e 00000 000 000 101 000 011 0001 0001 01 0111011110

m0f 00000 000 000 000 000 001 0000 0000 00 0110001111

m10 00000 000 000 100 000 001 0010 0000 00 0111011111 LDAX

m11 00000 000 000 111 000 011 0000 0000 00 0110011101

m12 00000 000 000 101 000 011 0001 0001 01 0111011110

m13 00000 000 000 000 000 001 0000 0000 00 0110001111

m14 00000 000 000 101 000 011 0001 0001 01 0111011110 LDA $K, X

m15 00000 000 000 101 000 011 0010 0000 00 0011011101

m16 00000 000 000 111 000 011 0000 0000 00 0110011101

m17 00000 000 000 101 000 011 0001 0001 01 0111011110

m18 00000 000 000 000 000 001 0000 0000 00 0110001111

m19 00000 000 000 101 000 011 0001 0001 01 0111011110 LDA ($K, X)

m1a 00000 000 000 111 000 001 0000 0000 00 0011011101

m1b 00000 000 000 101 000 011 0010 0000 00 0011011101

m1c 00000 000 000 111 000 011 0000 0000 00 0110011101

m1d 00000 000 000 101 000 011 0001 0001 01 0111011110

m1e 00000 000 000 000 000 001 0000 0000 00 0110001111

m1f 00000 000 000 101 000 011 0001 0001 01 0111011110 STA $K

m20 00000 000 000 111 000 001 0000 0000 00 0011011101

m21 00000 000 000 100 000 001 0000 0000 00 0000011111

m22 00000 000 000 101 000 011 0001 0001 01 0111011110

m23 00000 000 000 000 000 001 0000 0000 00 0110001111

MAPPER

m00 02 entolh LDA #K

m01 06 entolh LDX #K

m02 0a entolh LDA ($K)

m03 10 entolh LDAX

m04 14 entolh LDA $K, X

m05 19 entolh LDA ($K, X)

m06 1f entolh STA $K

MAIN

m00 00 opcode entolhs LDA #K

m01 09

m02 06 opcode entolhs STA $K

m03 30

m04 01 opcode entolhs LDX #K

m05 12

m06 02 opcode entolhs LDA ($K)

m07 80

m08 06 opcode entolhs STA $K

m09 32

m0a 03 opcode entolhs LDAX

m0b 06 opcode entolhs STA $K

m0c 34

m0d 04 opcode entolhs LDA $K, X

m0e 01

m0f 06 opcode entolhs STA $K

m10 36

m11 05 opcode entolhs LDA ($K, X)

m12 02

m13 06 opcode entolhs STA $K

m14 38

m18 05

m30 00 apotelesma LDA #K

m32 00 apotelesma LDA ($K)

m34 00 apotelesma LDAX

m36 00 apotelesma LDA $K, X

m38 00 apotelesma LDA ($K, X)

m80 83

m83 21