University of Patras

DEPARTMENT OF COMPUTER & INFORMATION ENGINEERING

COMPUTER ARCHITECTURE LABORATORY

EXERCISE 1st

The registers given in the workshop are as follows:

• Program Counter (PC): 00012

• Accumulator (ACC): 00002

• Auxiliary register X: 00102

**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 | 0001 | xx | x111010111 |
| m01 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

LDA $K: Load the Accumulator with the contents of address K (main memory)

1. 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.

2. MDR + 0 → X

The contents of the MDR are transferred to the X auxiliary register.

3. X + 0 → NOP, MAR

Addresses main memory with the contents of the X register, i.e. the number K.

4. MDR + 0 → ACC

The contents of the MDR, i.e. the contents of the K address, are transferred to 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 | 0001 | 0001 | 01 | x111011110 |
| m03 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 0010 | xx | x110011101 |
| m04 | xxxxx | 000 | xxx | 100 | 000 | 001 | 0010 | xxxx | xx | xx1101x1xx |
| m05 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 0000 | xx | x110011101 |
| m06 | xxxxx | 000 | xxx | 101 | 000 | 011 | 0001 | 0001 | 01 | x111011110 |
| m07 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

STA $K: Stores 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, which is the hexadecimal number K.

MDR + 0 → X

The contents of the MDR are transferred to the X auxiliary register.

X + 0 → NOP, MAR

Addresses main memory with the contents of the X register, i.e. the 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 |
| m08 | xxxxx | 000 | xxx | 101 | 000 | 011 | 0001 | 0001 | 01 | x111011110 |
| m09 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 0010 | xx | x110011101 |
| m0a | xxxxx | 000 | xxx | 100 | 000 | 001 | 0010 | xxxx | xx | xx1101x1xx |
| m0b | xxxxx | 000 | xxx | 100 | 000 | 001 | 0000 | xxxx | xx | xx10011101 |
| m0c | xxxxx | 000 | xxx | 101 | 000 | 011 | 0001 | 0001 | 01 | x111011110 |
| m0d | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**ADD $K: Adds the contents of address K to 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 → X**

**Addresses main memory with the contents of the X register, i.e. the number K.**

**X + 0 → NOP, MAR**

**Addresses main memory with the contents of the X register, i.e. the number K.**

**MDR + ACC → ACC**

**The result of adding the contents of the MDR, i.e. the contents of address K, and the contents of the Accumulator is 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.**

1. NEXT(PC)

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

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |
| m0e | xxxxx | 000 | xxx | 101 | 000 | 011 | 0001 | 0001 | 01 | x111011110 |
| m0f | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 0010 | xx | x110011101 |
| m10 | xxxxx | 000 | xxx | 100 | 000 | 001 | 0010 | xxxx | xx | xx1101x1xx |
| m11 | xxxxx | 000 | xxx | 101 | 000 | 011 | 0000 | 0000 | xx | x110011101 |
| m12 | xxxxx | 000 | xxx | 101 | 000 | 011 | 0001 | 0001 | 01 | x111011110 |
| m13 | 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 0010000000

m02 00000 000 000 101 000 011 0001 0001 01 0111011110 // LDA $K

m03 00000 000 000 111 000 011 0000 0010 00 0110011101

m04 00000 000 000 100 000 001 0010 0000 00 0111011111

m05 00000 000 000 111 000 011 0000 0000 00 0110011101

m06 00000 000 000 101 000 011 0001 0001 01 0111011110

m07 00000 000 000 000 000 001 0000 0000 00 0010000000

m08 00000 000 000 101 000 011 0001 0001 01 0111011110 // ADD $K

m09 00000 000 000 111 000 011 0000 0010 00 0110011101

m0a 00000 000 000 100 000 001 0010 0000 00 0111011111

m0b 00000 000 000 101 000 011 0000 0000 00 0110011101

m0c 00000 000 000 101 000 011 0001 0001 01 0111011110

m0d 00000 000 000 000 000 001 0000 0000 00 0010000000

m0e 00000 000 000 101 000 011 0001 0001 01 0111011110 // STA $K

m0f 00000 000 000 111 000 011 0000 0010 00 0110011101

m10 00000 000 000 100 000 001 0010 0000 00 0111011111

m11 00000 000 000 100 000 001 0000 0000 00 0100011111

m12 00000 000 000 101 000 011 0001 0001 01 0111011110

m13 00000 000 000 000 000 001 0000 0000 00 0010000000

MAPPER

m00 02 // LDA $K

m01 08 // ADD $K

m02 0e // STA $K

MAIN

m00 00

m01 08

m02 01

m03 09

m04 02

m05 0a

m06 f0

m07 ff

m08 03 //perioxh dedomenwn

m09 02

m0a 01

* The command merges that can be done in the applet of each macro are as follows:
* • In the macro LDA $K we can merge the macros MDR + 0 → X and X + 0 → MAR into MDR + 0 → NOP, MAR.
* The contents of the MDR instead of being first stored in an X register and then passed to the MAR can be passed directly from the MDR to the MAR. Likewise for the STA $K macro the MDR + 0 → X and X + 0 → NOP, MAR microinstructions can be merged into the MDR + 0 → NOP, MAR microinstruction.
* • Finally for the ADD $K macro the microinstructions MDR + 0 → X and can be consolidated into the microinstruction MDR + 0 → NOP, MAR.

Note that the auxiliary register X, in each of the microprograms of the macros can be omitted, as its use is not necessary. This way we reduce applet complexity and runtime by using fewer registers.