University of Patras

DEPARTMENT OF COMPUTER & INFORMATION ENGINEERING

COMPUTER ARCHITECTURE LABORATORY

EXERCISE 3rd

The registers given in the workshop are as follows:

• Program Counter (PC): 10102

• Accumulator (ACC): 11002

• Auxiliary X register: 11102

**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 | 1010 | xx | x111010111 |
| m01 | 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 |
| m02 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m03 | xxxxx | 000 | xxx | 101 | 000 | 001 | 1110 | xxxx | xx | xx11011101 |
| m04 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1100 | xx | x110011101 |
| m05 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m06 | 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.**

**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 |
| m07 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m08 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1110 | xx | x110011101 |
| m09 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m0a | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**INX: Increment the contents of the X auxiliary register by one.**

**X + 1 → X**

**The content of the auxiliary register X is incremented by one.**

**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 |
| m0b | xxxxx | 000 | xxx | 101 | 000 | 011 | 1110 | 1110 | 01 | x110011110 |
| m0c | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m0d | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**CMPX #Y: Compare the contents of auxiliary register X with the hexadecimal number Y.**

**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 Y.**

**MDR − X → NOP, MSTATUS**

**The content of the MDR, i.e. the number Y, is compared with the content of the auxiliary register X and the values ​​of the micro-flags are locked into the macro-flags.**

**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 |
| m0e | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m0f | xxxxx | 000 | xxx | 101 | 010 | 001 | 1110 | xxxx | xx | xx10111101 |
| m10 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m11 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**STA $K, X: ​​Store Accumulator in memory location with address R(X)+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 + X → NOP, MAR**

**The result of adding MDR, i.e. K, with the contents of the X register addresses the main memory.**

**ACC + 0 → NOP, MWE~**

**The contents of the Accumulator are stored at the main memory address pointed to by MAR, i.e. the address 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 |
| m12 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m13 | xxxxx | 000 | xxx | 101 | 000 | 001 | 1110 | xxxx | xx | xx11011101 |
| m14 | xxxxx | 000 | xxx | 100 | 000 | 001 | 1100 | xxxx | xx | xx0001x1xx |
| m15 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m16 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**ADC $K, X: ​​Add the Accumulator with the contents of the memory location addressed R(X)+K using latch and store the result in 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 + X → NOP, MAR**

**The result of adding MDR, i.e. K, with the contents of the X register addresses the main memory.**

**MDR + ACC → ACC, MSTATUS, CARRYE~**

**The result of adding the MDR, i.e. the contents of the memory location with address equal to R(X)+K, and the Accumulator is stored in the Accumulator and at the same time the CARRYE~ signal is activated so that the ALU performs an operation with an input latch.**

**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 |
| m17 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m18 | xxxxx | 000 | xxx | 101 | 000 | 001 | 1110 | xxxx | xx | xx11011101 |
| m19 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1100 | 1100 | xx | x110011001 |
| m1a | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m1b | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**CRC: Clear the prisoner flag (C=0).**

**0 + 0 → NOP, MSTATUS**

**The operation 0+0 is performed, which sets the value of the microC flag equal to zero, and activates the MSTATUS signal so that the values ​​of the micro-flags are locked to the macro-flags.**

**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 |
| m1c | xxxxx | 000 | xxx | 111 | 000 | 001 | xxxx | xxxx | 00 | xx10111110 |
| m1d | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m1e | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**BRNZ $K: Branch to address K (address K will be the new value of the program counter) if the zero flag is clear (Z=0).**

**BRZ(04)**

**Checks the macroZ flag. If its value is equal to unity then it goes to where the BRA(4:0) field specifies, i.e. 4 microinstructions below m23, to continue execution of the next macro, otherwise it continues with the serial execution of instructions.**

**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 → PC, MAR**

**The content of the MDR, i.e. the number K, is stored in the PC and addresses the main memory with the new value to perform another iteration.**

**NEXT (PC) µPC gets value from Mapper. Execution of the next macro from the appropriate applet continues.**

**PC +2 → PC, MAR**

**PC is incremented by two to terminate execution of the iterative process, and main memory is addressed 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 | 00100 | 011 | 011 | xxx | xxx | 001 | xxxx | xxxx | xx | xx1001x1xx |
| m20 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m21 | xxxxx | 000 | xxx | 111 | 000 | 011 | xxxx | 1010 | xx | xx11011101 |
| m22 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |
| m23 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 10 | x111011110 |
| m24 | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**SHLA: Left slip in contents of ACC.**

**ACC + 0 → ACC, SRAMU**

**The contents of the Accumulator are shifted one position to the left, that is, their original value is multiplied by two, and the result is 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 |
| m1c | xxxxx | 000 | xxx | 100 | 000 | 111 | 1100 | 1100 | xx | 011001x1xx |
| m1d | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 01 | x111011110 |
| m1e | xxxxx | 000 | xxx | xxx | xxx | 001 | xxxx | xxxx | xx | xx1000xxxx |

**HALT: End of program execution.**

**1. PC + 0 → PC, MAR**

**The Mapper continuously takes the HALT opcode as input.**

**2. 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 |
| m25 | xxxxx | 000 | xxx | 101 | 000 | 011 | 1010 | 1010 | 00 | x111011110 |
| m26 | 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, X

m03 00000 000 000 101 000 001 0010 0000 00 0011011101

m04 00000 000 000 111 000 011 0000 0000 00 0110011101

m05 00000 000 000 101 000 011 0001 0001 01 0111011110

m06 00000 000 000 000 000 001 0000 0000 00 0110001111

m07 00000 000 000 101 000 011 0001 0001 01 0111011110 LDX #K

m08 00000 000 000 111 000 011 0000 0010 00 0110011101

m09 00000 000 000 101 000 011 0001 0001 01 0111011110

m0a 00000 000 000 000 000 001 0000 0000 00 0110001111

m0b 00000 000 000 101 000 011 0010 0010 01 0110011110 INX

m0c 00000 000 000 101 000 011 0001 0001 01 0111011110

m0d 00000 000 000 000 000 001 0000 0000 00 0110001111

m0e 00000 000 000 101 000 011 0001 0001 01 0111011110 CMPX #Y

m0f 00000 000 000 101 010 001 0010 0000 00 0010111101

m10 00000 000 000 101 000 011 0001 0001 01 0111011110

m11 00000 000 000 000 000 001 0000 0000 00 0110001111

m12 00000 000 000 101 000 011 0001 0001 01 0111011110 STA $K, X

m13 00000 000 000 101 000 001 0010 0000 00 0111011101

m14 00000 000 000 100 000 001 0000 0000 00 0000011111

m15 00000 000 000 101 000 011 0001 0001 01 0111011110

m16 00000 000 000 000 000 001 0000 0000 00 0110001111

m17 00000 000 000 101 000 011 0001 0001 01 0111011110 ADC $K, X

m18 00000 000 000 101 000 001 0010 0000 00 0111011101

m19 00000 000 000 101 000 011 0000 0000 00 0110011001

m1a 00000 000 000 101 000 011 0001 0001 01 0111011110

m1b 00000 000 000 000 000 001 0000 0000 00 0110001111

m1c 00000 000 000 111 000 001 0000 0000 00 0010111110 CRC

m1d 00000 000 000 101 000 011 0001 0001 01 0111011110

m1e 00000 000 000 000 000 001 0000 0000 00 0110001111

m1f 00100 011 011 111 000 001 0000 0000 00 0010011110 BRNZ $K

m20 00000 000 000 101 000 011 0001 0001 01 0111011110

m21 00000 000 000 111 000 011 0000 0001 00 0111011101

m22 00000 000 000 000 000 001 0000 0000 00 0110001111

m23 00000 000 000 101 000 011 0001 0001 10 0111011110

m24 00000 000 000 000 000 001 0000 0000 00 0110001111

m25 00000 000 000 101 000 011 0001 0001 00 0111011110 HALT

m26 00000 000 000 000 000 001 0000 0000 00 0110001111

m27 00000 000 000 100 000 111 0000 0000 00 0110011111 SHLA

m28 00000 000 000 101 000 011 0001 0001 01 0111011110

m29 00000 000 000 000 000 001 0000 0000 00 0110001111

MAPPER

m00 02 entolh LDA $K,X

m01 07 entolh LDX #K

m02 0b entolh INX

m03 0e entolh CMPX #Y

m04 12 entolh STA $K,X

m05 17 entolh ADC $K,X

m06 1c entolh CRC

m07 1f entolh BRNZ $K

m08 25 entolh HALT

m09 27 entolh SHLA

MAIN

m00 01 opcode LDX #K

m01 00

m02 06 opcode CRC

m03 00 opcode LDA $K, X

m04 10

m05 09 opcode SHLA

m06 05 opcode ADC #K, X

m07 20

m08 04 opcode STA $K, X

m09 30

m0a 02 opcode INX

m0b 03 opcode CMPX #Y

m0c 08

m0d 07 opcode BRNZ $K

m0e 02

m0f 08 opcode HALT

m10 01 periexomeno z

m11 02

m12 03

m13 04

m14 05

m15 06

m16 07

m17 08

m20 09 periexomeno y

m21 08

m22 07

m23 06

m24 05

m25 04

m26 03

m27 02

m30 00 periexomeno w

m31 00

m32 00

m33 00

m34 00

m35 00

m36 00

m37 00