Designing the processor

Datapath

Click here

Instruction set

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction** | **Opcode** | **Operand (16 bit)** | **Operation** |
| loadac | 0000 0100 | - | ac <= DM[ac] |
| movacr | 0000 1000 | - | r <= ac |
| movacr1 | 0000 1001 | - | r1 <= ac |
| movacr2 | 0000 1010 | - | r2 <= ac |
| movacr3 | 0000 1011 | - | r3 <= ac |
| movacr4 | 0000 1100 | - | r4 <= ac |
| movacr5 | 0000 1101 | - | r5 <= ac |
| movacdar | 0000 1110 | - | dar <= ac |
| movrac | 0000 1111 | - | ac <= r |
| movr1ac | 0001 0000 | - | ac <= r1 |
| movr2ac | 0001 0001 | - | ac <= r2 |
| movr3ac | 0001 0010 | - | ac <= r3 |
| movr4ac | 0001 0011 | - | ac <= r4 |
| movr5ac | 0001 0100 | - | ac <= r5 |
| movdarac | 0001 0101 | - | ac <= dar |
| stac | 0001 0110 | - | DM[ac] <= ac |
| add | 0001 1001 | - | ac <= ac + r |
| sub | 0001 1011 | - | ac <= ac – r |
| lshift | 0001 1101 | - | ac <= r << ac |
| rshift | 0001 1111 | - | ac <= r >> ac |
| incac | 0010 0001 | - | ac <= ac + 1 |
| incdar | 0010 0010 | - | dar <= dar +1 |
| incr1 | 0010 0011 | - | r1 <= r1 +1 |
| incr2 | 0010 0100 | - | r2 <= r2 +1 |
| incr3 | 0010 0101 | - | r3 <= r3 + 1 |
| loadim | 0010 0110 | imm | ac <= imm |
| jumpz | 0010 1001 | imm | if z == 1, go to instruction IM[imm] |
| jumpnz | 0011 0000 | imm | if z == 0, go to instruction IM[imm] |
| jump | 0011 0001 | imm | go to instruction IM[imm] |
| nop | 0011 0010 | - | no operation |
| endop | 0011 0011 | - | end process |

Microinstructions

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction (8bits)** | **Opcode**  **(decimal)** | **Microinstruction (6bits)** | **Operation** |
| idle  (internal) | - | idle | goto fetch1 if status == 1 else idle |
| fetch  (internal) | - | fetch1 | read IMEM |
| write ir |
| goto fetch2 |
| fetch2 | read IMEM |
| write ir |
| goto fetch3 |
| fetch3 | read IMEM |
| goto ir[5:0] |
| loadac | 4 | loadac1 | read ac |
| write dar |
| goto loadac2 |
| loadac2 | read ac |
| wrirte dar |
| goto loadac3 |
| loadac3 | read DMEM |
| write ac |
| goto loadac4 |
| loadac4 | read DMEM |
| write ac |
| increment pc |
| goto fetch1 |
| movacr | 8 | movacr | read ac |
| write r |
| increment pc |
| goto fetch1 |
| movacr1 | 9 | moveacr1 | read ac |
| write r1 |
| increment pc |
| goto fetch1 |
| movacr2 | 10 | moveacr2 | read ac |
| write r2 |
| increment pc |
| goto fetch1 |
| movacr3 | 11 | moveacr3 | read ac |
| write r3 |
| increment pc |
| goto fetch1 |
| movacr4 | 12 | moveacr4 | read ac |
| write r4 |
| increment pc |
| goto fetch1 |
| movacr5 | 13 | moveacr5 | read ac |
| write r5 |
| increment pc |
| goto fetch1 |
| movacdar | 14 | moveacdar | read ac |
| write dar |
| increment pc |
| goto fetch1 |
| movrac | 15 | moverac | read r |
| write ac |
| increment pc |
| goto fetch1 |
| movr1ac | 16 | mover1ac | read r1 |
| write ac |
| increment pc |
| goto fetch1 |
| movr2ac | 17 | mover2ac | read r2 |
| write ac |
| increment pc |
| goto fetch1 |
| movr3ac | 18 | mover3ac | read r3 |
| write ac |
| increment pc |
| goto fetch1 |
| movr4ac | 19 | mover4ac | read r4 |
| write ac |
| increment pc |
| goto fetch1 |
| movr5ac | 20 | mover5ac | read r5 |
| write ac |
| increment pc |
| goto fetch1 |
| movdarac | 21 | movedarac | read dar |
| write ac |
| increment pc |
| goto fetch1 |
| stac | 22 | stac1 | read ac |
| goto stac2 |
| stac2 | read ac |
| write dm |
| goto stac3 |
| stac3 | read ac |
| increment pc |
| goto fetch1 |
| add | 25 | add1 | add alu |
| goto add2 |
| add2 | add alu |
| write alu\_to\_ac |
| increment pc |
| goto fetch1 |
| sub | 27 | sub1 | sub alu |
| goto sub2 |
| sub2 | sub alu |
| write alu\_to\_ac |
| increment pc |
| goto fetch1 |
| lshift | 29 | lshift1 | lshift alu |
| goto lshift2 |
| lshift2 | lshift alu |
| write alu\_to\_ac |
| increment pc |
| goto fetch1 |
| rshift | 31 | rshift1 | rshift alu |
| goto rshift2 |
| rshift2 | rshift alu |
| write alu\_to\_ac |
| increment pc |
| goto fetch1 |
| incac | 33 | incac | increment ac |
| incremtnt pc |
| goto fetch1 |
| incdar | 34 | incdar | increment dar |
| incremtnt pc |
| goto fetch1 |
| incr1 | 35 | incr1 | increment r1 |
| incremtnt pc |
| goto fetch1 |
| incr2 | 36 | incr2 | increment r2 |
| incremtnt pc |
| goto fetch1 |
| incr3 | 37 | incr3 | increment r3 |
| incremtnt pc |
| goto fetch1 |
| loadim | 38 | loadim1 | increment pc |
| goto loadim2 |
| loadim2 | read IMEM |
| goto loadim3 |
| loadim3 | read IMEM |
| write ac |
| increment pc |
| goto fetch1 |
| jump | 49 | jump | increment pc |
| goto jumpz2 |
| jumpz | 41 | jumpz1 | increment pc |
| goto jumpz2 if z == 1 else jumpz6 |
| jumpz2 | read IMEM |
| write pc |
| goto jumpz3 |
| jumpz3 | read IMEM |
| write pc |
| goto jumpz4 |
| jumpz4 | read IMEM |
| write ir |
| goto jumpz5 |
| jumpz5 | read IMEM |
| write ir |
| goto fetch3 |
| jumpz6 | goto jumpz7 |
| jumpz7 | increment pc |
| goto jumpz4 |
| jumpnz | 48 | jumpnz1 | increment pc |
| goto jumpz3 if z == 0 else goto jumpz7 |
| endop | 51 | endop | read DMEM |
| goto endop |
| nop | 50 | nop | increment pc |
| goto fetch1 |

Finite state machines for microinstructions

Click here

Modules and Components

Diagram, schematic

Description automatically generatedMachine

|  |  |
| --- | --- |
| Inputs | Outputs |
| Clk – clock | End\_process – signal that indicate process has ended |
| Status – control signal for change phases (input image data, process data, output data) | Data\_out – ouput image data |
| Data\_in – input image data |  |
| Data\_addr\_in – input data memory address |  |

Inside of the machine

Processor

Chart, diagram

Description automatically generated

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Inputs | Outputs |
| Clk – clock | End\_process – signal that indicate process has ended |
| Status – control signal for change phases (input image data, process data, output data) | Dm\_en – data memory write enable signal |
| Dm\_out – output of data memory | Im\_en – instruction memory write enable signal |
| Im\_out – output of instruction memory | Pc\_out – instruction memory address |
| Data\_in – Input image data | Dar\_out – data memory address |
| Data\_addr\_in – input data memory address | Bus\_out – data output for data memory |

Modules inside the processor

Arithmetic & Logic Unit (ALU) – Do arithmetic and logic operations in the processor

Control Unit – Do all the controlling in the processor. Implemented as a finite state machine.

Special purpose registers

Accumulator (ac) – input for the ALU, store ALU output

Program counter (pc) – directed to the next instruction to be fetched

Instruction register (ir) – store the instruction fetched from the instruction memory

Data address register (dar) – keep the address of the store location of the data memory. Loading from data memory and storing to data memory are being done using this address

General purpose registers

R1, r2, r3 – general purpose registers which can be incremented

R4, r5 – general purpose registers which cannot be incremented

ALU

A picture containing text, antenna

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Alu\_op | Operation | Description |
|  | Addition | Alu\_out <= in1 + in2 |
|  | Subtraction | Alu\_out <= in2 – in1 |
|  | Left shift | Alu\_out <= in1 << in2 |
|  | Right shift | Alu\_out <= in1 >> in2 |

Z is 1 if the alu\_out is zero. Z is 0 if the alu\_out is nonzero.

Control