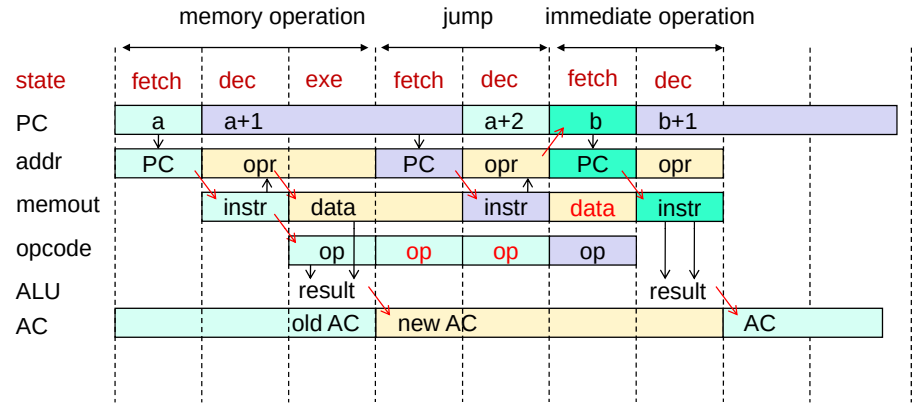
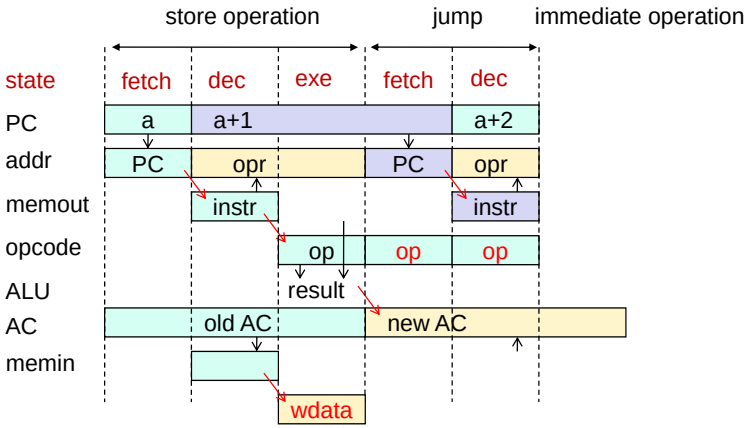
CPU 설계

Design Example:
CPU

타이밍

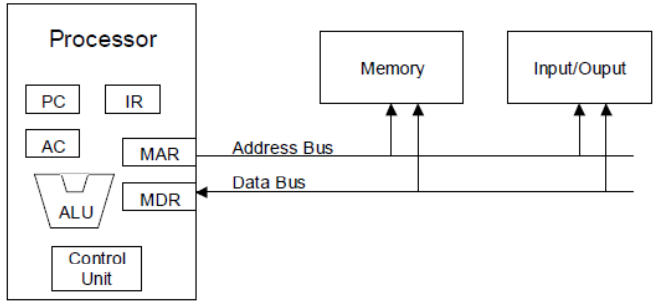


타이밍



Simple Computer Design: μP3

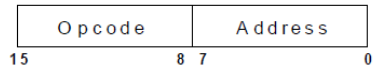
- simple computer system의 구조



- μP3 processor
  - accumulator 기반 프로세서

μP3 Instructions and Instruction Format

- Instruction Format
  - 16-bit length



- Instructions

Instruction Mnemonic		Operation Preformed	Opcode Value
ADD	*address*	AC <= AC + contents of memory address	00
STORE	*address*	contents of memory address <= AC	01
LOAD	*address*	AC <= contents of memory address	02
JUMP	*address*	PC <= address	03
JNEG	*address*	If AC < 0 Then PC <= address	04

Example Program

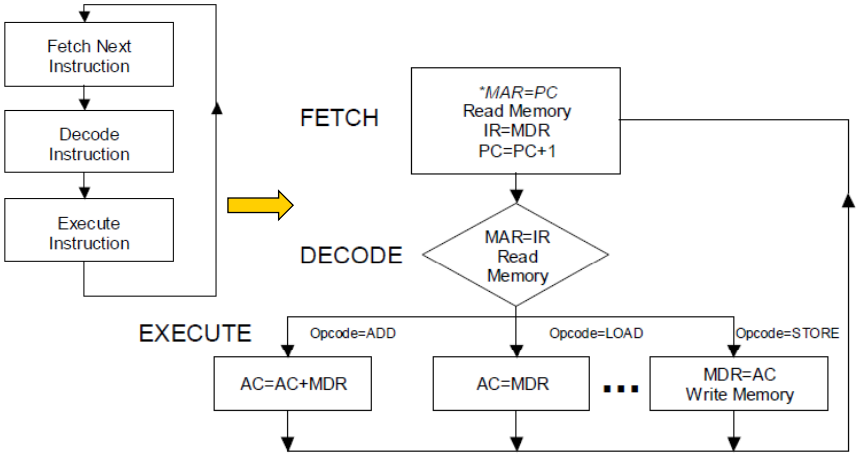
- Computer Program for A = B + C
  - 변수 A, B, C가 각각 메모리 10h, 11h, 12h번지에 저장됨

Assembly Language	Machine Language
LOAD B	0211
ADD C	0012
STORE A	0110

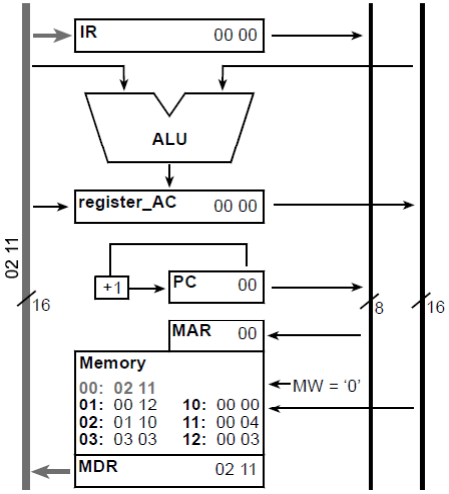
- Programmer visible registers
  - AC (accumulator)
  - PC (program counter)
- CPU internal registers
  - IR (instruction register)
  - MAR (memory address register)
  - MDR (memory data register)

Processor Cycles

- Processor Fetch, Decode, Execute Cycle

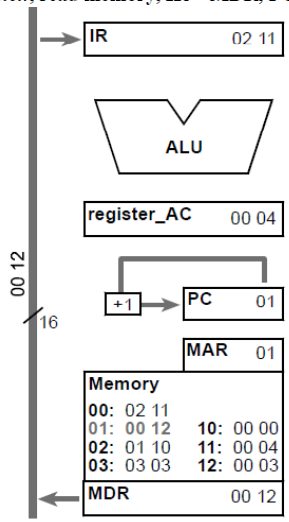


Datapath after reset



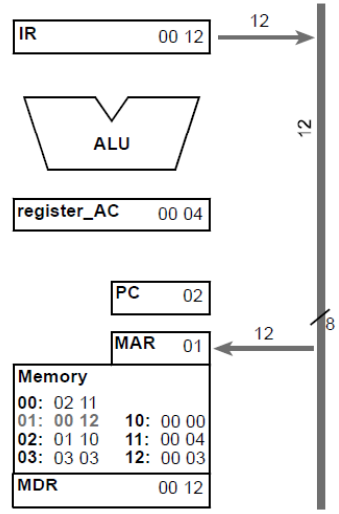
ADD instruction' Fetch state

MAR = PC prior to fetch, read memory, IR = MDR, PC = PC + 1



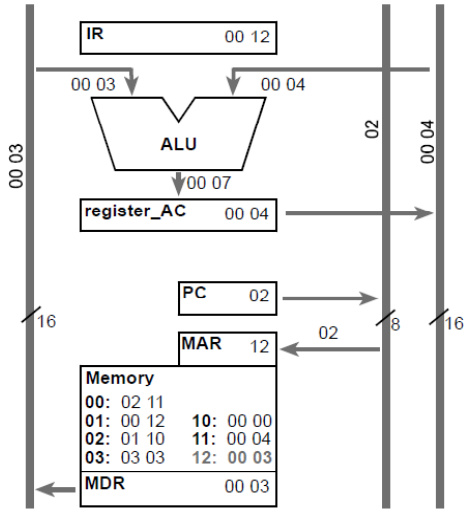
ADD instruction's Decode state

Decode Opcode to find Next State, MAR = IR, and start memory read



ADD instruction's Execute state

AC = AC + MDR, MAR = PC*, and GOTO FETCH



Altera FPGA internal RAM 사용

```
// Use Altsynram function for computer's memory (256 16-bit words)
altsyncram      altsyncram_component (
                                .wren_a (memory_write_out),
                                .clock0 (clock),
                                .address_a (memory_address_register_out),
                                .data_a (register_A),
                                .q_a (memory_data_register));
        defparam
                altsyncram_component.operation_mode = "SINGLE_PORT",
                altsyncram_component.width_a = 16,
                altsyncram_component.widthad_a = 8,
                altsyncram_component.outdata_reg_a = "UNREGISTERED",
                altsyncram_component.lpm_type = "altsyncram",
// Reads in mif file for initial program and data values
                altsyncram_component.init_file = "program.mif",
                altsyncram_component.intended_device_family = "Cyclone";
```

SRAM 초기화 파일: program.mif

```
DEPTH = 256;                % Memory depth and width are required      %
WIDTH = 16;                 % Enter a decimal number          %

ADDRESS_RADIX = HEX;        % Address and value radixes are optional   %
DATA_RADIX = HEX;           % Enter BIN, DEC, HEX, or OCT; unless      %
                            % otherwise specified, radixes = HEX       %

                    -- Specify values for addresses, which can be single address or range
CONTENT
  BEGIN
    [00..FF]    :   0000;   % Range--Every address from 00 to FF = 0000 (Default) %
      00    :       0210;   % LOAD AC with MEM(10) %
      01    :       0011;   % ADD MEM(11) to AC %
      02    :       0112;   % STORE AC in MEM(12) %
      03    :       0212;   % LOAD AC with MEM(12) check for new value of FFFF %
      04    :       0304;   % JUMP to 04 (loop forever) %
      10    :       AAAA;   % Data Value of B %
      11    :       5555;   % Data Value of C%
      12    :       0000;   % Data Value of A - should be FFFF after running program %
  END ;
```