# BUILDING A ROM WITH XILINX WEBPACK 6.X

1. Open your project.
2. Project -> New Source.
   a. Select VHDL module
   b. File Name: “InstROM”
   c. Select Next
   d. Port Name “InstAddress”, in, MSB=8,LSB=0 (for a 512-entry ROM)
   e. Port Name “InstOut”, out, MSB=7,LSB=0
   f. Select Next
   g. Select Finish
3. Edit the VHDL code in InstROM.vhd, so it looks like this:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity InstROM is
    Port ( InstAddress : in std_logic_vector(8 downto 0);
           InstOut : out std_logic_vector(7 downto 0));
end InstROM;

architecture Behavioral of InstROM is

    type ROM_Array is array (0 to 511)
      of std_logic_vector(7 downto 0);

    constant Content: ROM_Array := (
      X"12",
      X"23",
      X"45",
      X"FA",
      X"3E",
      X"93",
      X"12",
      X"32",
      X"99",
others => X"00"   );

begin

      InstOut <= Content(conv_integer(InstAddress));

end Behavioral;
```

This is for a 512-entry ROM (9-bit PC). The values in red are those that would change for a different sized ROM.
Your ROM contents will be different (and much longer!). This puts hex 0x12 in location 0, 0x23 in location 1, 0x45 in location 2, etc.

4. In “Proccesses for Source” window, click “Check Syntax” (in “Synthesize XST”) – correct errors.

5. In “Proccesses for Source” window, click “Create Schematic Symbol” (in “Design Entry Utilities”)
6. Open Schematic in which you want to include your ROM.
7. “instrom” should now be added to your library of symbols. You may want to edit its shape to make it look nicer.
8. Test it alone to make sure it is reading the right values, with the bits in the order you expect, etc.

## BUILDING A RAM WITH INITIALIZED VALUES WITH XILINX WEBPACK 6.X

- Open your project.
- Project -> New Source.
- Select VHDL module
    - File Name: “DataRAM”
    - Select Next
    - Port Name “DataAddress”, in, MSB=7,LSB=0 (for a 256-entry RAM)
    - Port Name “clk”, in
    - Port Name “ReadMem”, in
    - Port Name “WriteMem”, in
    - Port Name “DataIn”, in, MSB=7, LSB=0
    - Port Name “DataOut”, out, MSB=7,LSB=0
    - Select Next
    - Select Finish
- Edit the VHDL code DataRAM.vhd, so it looks like this:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity DataRAM is
    Port ( DataAddress : in std_logic_vector(7 downto 0);
           clk : in std_logic;
           ReadMem : in std_logic;
           WriteMem : in std_logic;
        DataIn : in std_logic_vector (7 downto 0);
           DataOut : out std_logic_vector(7 downto 0));
end DataRAM;

architecture Behavioral of DataRAM is

type ram_type is array (0 to 255) of
   std_logic_vector(7 downto 0);
signal tmp_ram: ram_type := (
   0 => X"A8",
   1 => X"D7",
   32 => X"39",
   33 => X"3A",
   34 => X"0F",
   35 => X"12",
   36 => X"17",
37 => X"32",
38 => X"12",
39 => X"2C",
40 => X"39",
41 => X"3A",
42 => X"0F",
43 => X"12",
44 => X"17",
45 => X"32",
46 => X"12",
47 => X"2C",
48 => X"39",
49 => X"3A",
50 => X"0F",
51 => X"12",
52 => X"17",
53 => X"32",
54 => X"12",
55 => X"2C",
56 => X"39",
57 => X"3A",
58 => X"0F",
59 => X"12",
60 => X"17",
61 => X"32",
62 => X"12",
63 => X"2C",
64 => X"40",
65 => X"41",
66 => X"42",
67 => X"43",
68 => X"44",
69 => X"45",
70 => X"46",
71 => X"47",
72 => X"48",
73 => X"49",
74 => X"4A",
75 => X"4B",
76 => X"4C",
77 => X"4D",
78 => X"4E",
79 => X"4F",
80 => X"50",
81 => X"51",
82 => X"52",
83 => X"53",
84 => X"54",
85 => X"55",
86 => X"56",
87 => X"57",
88 => X"58",
89 => X"59",
90 => X"5A",
91 => X"5B",
92 => X"5C",
93 => X"5D",
   94 => X"5E",
   95 => X"5F",

   others => X"00");

begin
    process(ReadMem, DataAddress)
    begin
         if ReadMem='1' then
             DataOut <= tmp_ram(conv_integer(DataAddress));
         else
             DataOut <= (DataOut'range => 'Z');
         end if;
    end process;

 process(clk, WriteMem)
    begin
   if (clk'event and clk='1') then
         if WriteMem='1' then
             tmp_ram(conv_integer(DataAddress)) <= DataIn;
         end if;
   end if;
end process;
end Behavioral;
```

---

This is for a 256-entry RAM (8-bit address). Your initial RAM contents will be smaller than what is shown here. You need to make 2 RAMs one for each input set (A and B).

- In "Proccesses for Source" window, click "Check Syntax" (in "Synthesize XST") – correct errors.
- In "Proccesses for Source" window, click "Create Schematic Symbol" (in "Design Entry Utilities")
- Open Schematic in which you want to include your RAM.
- "dataram" should now be added to your library of symbols. You may want to edit its shape to make it look nicer.
- Test it alone to make sure it is reading the right values, with the bits in the order you expect, etc.