این پروژه از چندین بخش تشکیل شده که به ترتیب به توضیح آن ها پرداخته میشود

microproseccor.1 : در این کد بدنه اصلیکار تعریف شده است ابتدا تمامی ورودی ها و خروجی های تعریف شده اند که عکس آن در زیر آمده است :

```
entity microprocessor is
    port
    (
        reset   :in std_logic;                  -- reset
        clk     :in std_logic;                  -- clock
        data    :inout std_logic_vector(15 downto 0);   -- data lines
        address :out std_logic_vector(11 downto 0); -- address lines
        memr    :out std_logic;                 -- memory read
        memw    :out std_logic;                 -- memory write
        inport  :in std_logic_vector(7 downto 0);       -- input port
        outport     :out std_logic_vector(7 downto 0);  -- output port
        intr_in :in std_logic;                  -- interrupt for input
        intr_out    :in std_logic                   -- interrupt for output
    );
end microprocessor;
```

سپس بدنه ساختاری آن تعریف شده است در ابتدای این قسمت تعدادی سگنال میانی تعریف شده اند که عکس آن در زیر آمده است :

```
architecture main of microprocessor is
                -- list of registers
    signal dr   :std_logic_vector(15 downto 0); -- data register
    signal ar   :std_logic_vector(11 downto 0); -- address register
    signal ac   :std_logic_vector(15 downto 0); -- accumulator
    signal ir       :std_logic_vector(15 downto 0); -- instruction register
    signal pc   :std_logic_vector(11 downto 0); -- program counter
    signal tr       :std_logic_vector(15 downto 0); -- temporary register
    signal inpr :std_logic_vector(7 downto 0);  -- input register
    signal outr :std_logic_vector(7 downto 0);  -- output register

                -- internals of microprocessor
    signal d        :std_logic_vector(7 downto 0);  -- instruction decoder output
    signal sc   :std_logic_vector(3 downto 0);  -- sequence counter output
    signal t        :std_logic_vector(15 downto 0); -- timing signals
    signal i        :std_logic;         -- I bit
    signal e        :std_logic;         -- extended accumulator
    signal s        :std_logic;         -- start-stop flip-flop
    signal en_id    :std_logic;         -- enable signal for instruction decoder
    signal clr_sc   :std_logic;         -- clear signal for sequence counter
    signal fgi  :std_logic;         -- input flag
    signal fgo  :std_logic;         -- output flag
    signal ien  :std_logic;         -- interrupt enable flip-flop
    signal r        :std_logic;         -- interrupt flip-flop
```

بعد از تعریف سیگنال های گفته شده برای قسمت های مختلف تعریف رفتاری آن آغاز میشود

در ابتدا در یک پروسس به تعریف یک دیکدر پایه پرداخته میشود و در پراسس بعدی توالی شمار مورد استفاده تعریف میشود که کد آنها در زیر آمده است :

```
begin
process(en_id)              -- instruction decoder
    begin
    if en_id='1' then
        case ir(14 downto 12) is
            when "000" => d <= "00000001";
            when "001" => d <= "00000010";
            when "010" => d <= "00000100";
            when "011" => d <= "00001000";
            when "100" => d <= "00010000";
            when "101" => d <= "00100000";
            when "110" => d <= "01000000";
            when "111" => d <= "10000000";
            when others => null;
        end case;
    end if;
end process;

process(clk,s,clr_sc)          -- 4-bit sequence counter
    begin
    if s='0' then
        if (clk'event and clk='1') then
            if clr_sc='1' then
                sc <= "0000";
            else
                sc <= sc + "0001";
            end if;
        end if;
    end if;
end process;
```

سپس یک دیکدر برای دیکد کردن مقدار توالی شمار تعریف گشته است.

بعد از آن عملکرد رفتاری توالی شمار تعریف شده است

سپس رفتار مدار با توجه به توضیحات فایل ارسالی انجام گرفته است(از خط 108 تا 311)

divder.2 : **در این بخش یک تقسیم کننده فرکانس تعریف شده که میتواند فرکانس تعریف شده که میتواند فرکانس کلاک ورودی را با توجه به خواسته ما تغییر دهد.**

ram.3 : در این کد یک حافظه مورد نیاز با توجه به خواسته مسئله تعریف شده است.ابتدا ورودی ها و خروجی ها مانند دو مثال قبل که توضیح داده شد تعریف و مشخص شده اند سپس در قسمت ساختار یک نوع جدید از داده(با استفاده از دستور type) تعریف شده است که برای شناساندن رم میباشد رم دارای یک خط خواندن و یک نوشتن هست که برای عملیات هایم مختلف مورد استفاده قرار میگیرد.

ebscu.4 : این قسمت واحد کنترلی حافظه میباشد و عبارتی کنترل میکند که کی رم در حال خواندن است و کی نوشتن تا بطور اتفاقی وقتی رم مشغول خواندن است عمل نوشتن صورت نگیرد و بعکس.

testbech.5 : در نهایت برای تست عملکرد این مدار یک تست نوشته شده است که در آن حالت های مختلف بررسی میشود با اجرای این کد شکل موج های مختلف خروجی و توجه به ورودی های مختلف قابل مشاهده میباشد