    assign Rom[5'h00]=32'h20010008;

    //*001000 00000 00001 0000000000001000 addi $1,$0,8 ->$1=8*

    assign Rom[5'h01]=32'h3402000C;

    //*001101 00000 00010 0000000000001100 ori $2,$0,12 ->$2=12*

    assign Rom[5'h02]=32'h00221820;

    //*000000 00001 00010 00011 00000 100000 add $3,$1,$2 ->$3=20*

    //*此时发生数据冒险，$1来自第1条指令的WB前推；$2来自第2条指令的MEM级前推*

    assign Rom[5'h03]=32'h00412022;

    //*000000 00010 00001 00100 00000 100010 sub $4,$2,$1->$4=4*

    //*此时发生数据冒险，$2来自第2条指令的WB级前推*

    assign Rom[5'h04]=32'h00222824;

    //*000000 00001 00010 00101 00000 100100 and $5,$1,$2->$5=8*

    assign Rom[5'h05]=32'h00223025;

    //*000000 00001 00010 00110 00000 100101 or $6,$1,$2->$6=12*

    assign Rom[5'h06]=32'h14220006;

    //*000101 00001 00010 000000000110 bne $1,$2,6*

    //*由于$1,$2不相等，会等待两个时钟周期后跳转到Rom[5'h0D]指令*

    //*此时出现结构冒险，缩短分支的延迟*

    assign Rom[5'h07]=32'h00221820;

    //*add $3,$1,$2 由于上面的结构冒险，此条指令会被读取，然后被清零，不执行*

    assign Rom[5'h08]=32'h00412022;

    //*sub $4,$2,$1 由于上面的结构冒险，此条指令会被读取，然后被清零，不执行*

    assign Rom[5'h09]=32'hXXXXXXXX;

    assign Rom[5'h0A]=32'hXXXXXXXX;

    assign Rom[5'h0B]=32'hXXXXXXXX;

    assign Rom[5'h0C]=32'hXXXXXXXX;

    assign Rom[5'h0D]=32'hAD02000A;

    //*101011 01000 00010 0000000000001010 sw $2 10($8)->memory[$8+10]=12*

    //*即存储器编号为2的值改变为12*

    assign Rom[5'h0E]=32'h8D04000A;

    //*100011 01000 00100 0000000000001010 lw $4 10($8)->$4=memory[$8+10]=12*

    //*此时发生lw数据冒险*

    assign Rom[5'h0F]=32'h10240006;

    //*000100 00001 00100 0000000000000110 beq $1,$4,6*

    //*由于发生lw数据冒险，会暂停一个周期，且$4来自前一条指令的WB级数据前推*

    //*由于$1,$4不相等，则不发生转移，也不清除后续两条指令*

    assign Rom[5'h10]=32'h30470009;

    //*001100 00010 00111 0000000000001001 andi $7,$2,9->$7=8*

    assign Rom[5'h11]=32'h382300EF;

    //*001110 00001 00011 0000000011101111 xori $3,$1,0xef->$3=0xe7*

    assign Rom[5'h12]=32'h3C011234;

    //*001111 00000 00001 0001001000110100 lui $1,0x1234->$1=0x12340000*

    assign Rom[5'h13]=32'h00021900;

    //*000000 00000 00010 00011 00100 000000 sll $3,$2,4->$3=0xC0*

    assign Rom[5'h14]=32'h00011102;

    //*000000 00000 00001 00010 00100 000010 srl $2,$1,4->$2=0x01234000*

    assign Rom[5'h15]=32'h00020903;

    //*000000 00000 00010 00001 00100 000011 sra $1,$2,4->$1=0x00123400*

    //*此时发生数据冒险，$2来自上一条指令MEM级的前推*

    assign Rom[5'h16]=32'h0C00001A;

    //*000011 00000000000000000000011010 Jal 1A*

    //*跳转到Rom[5'h1A]指令，并将pc+4的值写入31号寄存器*

    assign Rom[5'h17]=32'h0800001D;

    //*在执行上一条指令时，这条指令读取不执行*

    //*000010 00000000000000000000011101 J 1D*

    //*跳转到Rom[5'h1D]指令*

    assign Rom[5'h18]=32'h00021900;//*sll $3,$2,4*

    //*该指令会被读取，但不会执行*

    assign Rom[5'h19]=32'h00011102;//*srl $2,$1,4*

    //*该指令会被读取，但不会执行*

    assign Rom[5'h1A]=32'h00437826;

    //*000000 00010 00011 01111 00000 100110 xor $15,$2,$3->$15=0x012340C0*

    assign Rom[5'h1B]=32'h00021903;

    //*000000 00000 00010 00011 00100 000011 sra $3,$2,4->$3=0x00123400*

    assign Rom[5'h1C]=32'h03E00008;

    //*000000 11111 00000 00000 00000 001000 Jr 17*

    //*由于31号寄存器保存了17,这里直接跳转到Rom[5'h17]指令*

    assign Rom[5'h1D]=32'h00020903;//*sra $1,$2,4*

    assign Rom[5'h1E]=32'h00020903;//*sra $1,$2,4*

    assign Rom[5'h1F]=32'h00020903;//*sra $1,$2,4*