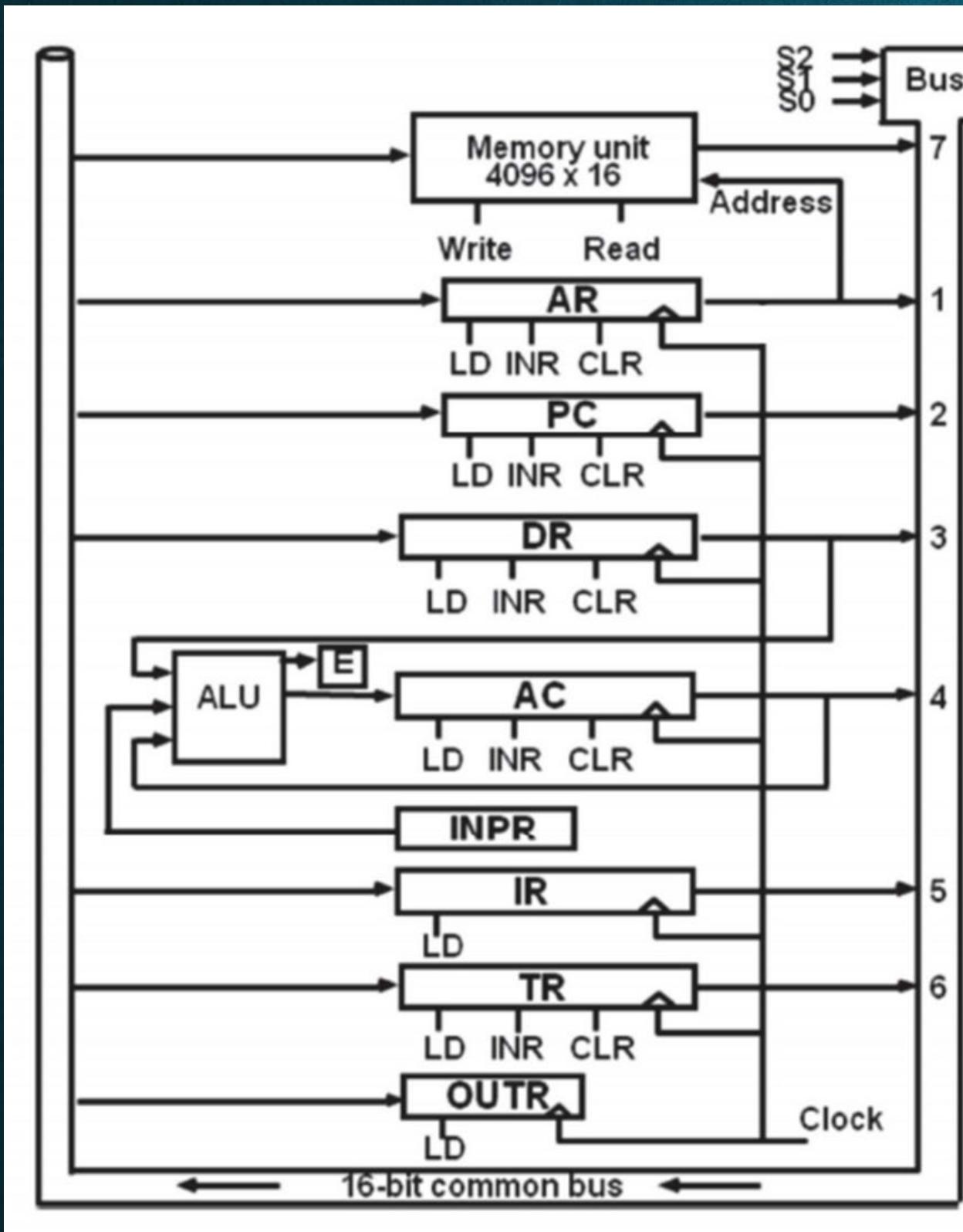


MANO BASIC COMPUTER PROJECT



Eslam ahmed hosny
Mohamed Emad Abdelgillil
Basil Magdy Shrif

INTRODUCTION



Register → STD_logic_vector
Flip flops → STD_logic
RAM -> array

Register → STD_logic_vector

```
-- Main Registers
signal PC  : STD_LOGIC_VECTOR(11 downto 0) := "000000000010";  -- PC = 2
signal AR  : STD_LOGIC_VECTOR(11 downto 0) := (others => '0');
signal IR  : STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
signal DR  : STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
signal AC  : STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
signal OUTR_out: STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
signal OUTR_in: STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
signal INPR_in: STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
signal INPR_out: STD_LOGIC_VECTOR(7 downto 0) := (others => '0');
signal TR  : STD_LOGIC_VECTOR(15 downto 0) := (others => '0');
```

Main CPU Process

```
process(clk, reset)
  variable temp_add : unsigned(16 downto 0);
begin
```

USE TR VARIABLE INSTEAD FOR CARRY

OVERVIEW

Flip flops → STD_logic

-- Flip-Flops

```
signal I    : STD_LOGIC := '0';
signal E    : STD_LOGIC := '0';
signal R    : STD_LOGIC := '0';
signal IEN  : STD_LOGIC := '0';
signal FGI  : STD_LOGIC := '0';
signal FGO  : STD_LOGIC := '0';
```

RAM -> array

```
type mem_type is array(0 to 4095) of STD_LOGIC_VECTOR(15 downto 0);
signal RAM : mem_type := (

    0  => x"0000",
    1  => x"4032",    -- (Example: Initialization or unused)
    2  => x"203E",    -- LDA (Load Accumulator)

=> x"7001", ----HLT
-- Interrupt codes
50  => x"F800",
51  => x"F400",
52  => x"F200",
53  => x"F100",
54  => x"F080",    -- Duplicate value, but index is unique
55  => x"4000",    -- Moved from conflicting index 9

-- Data for testing
60  => x"00FF",    -- Data for AND
61  => x"0001",    -- Data for ADD
62  => x"00AA",    -- Data for LDA
63  => x"5100",    -- BSA
256 => x"0000",    -- Store PC -- 100
257 => x"6130",    -- 101 = PC
259 => x"7001",    -- HLT (Halt)
304 => x"FFFF",

others => (others => '0')
```

OVERVIEW



OVERVIEW

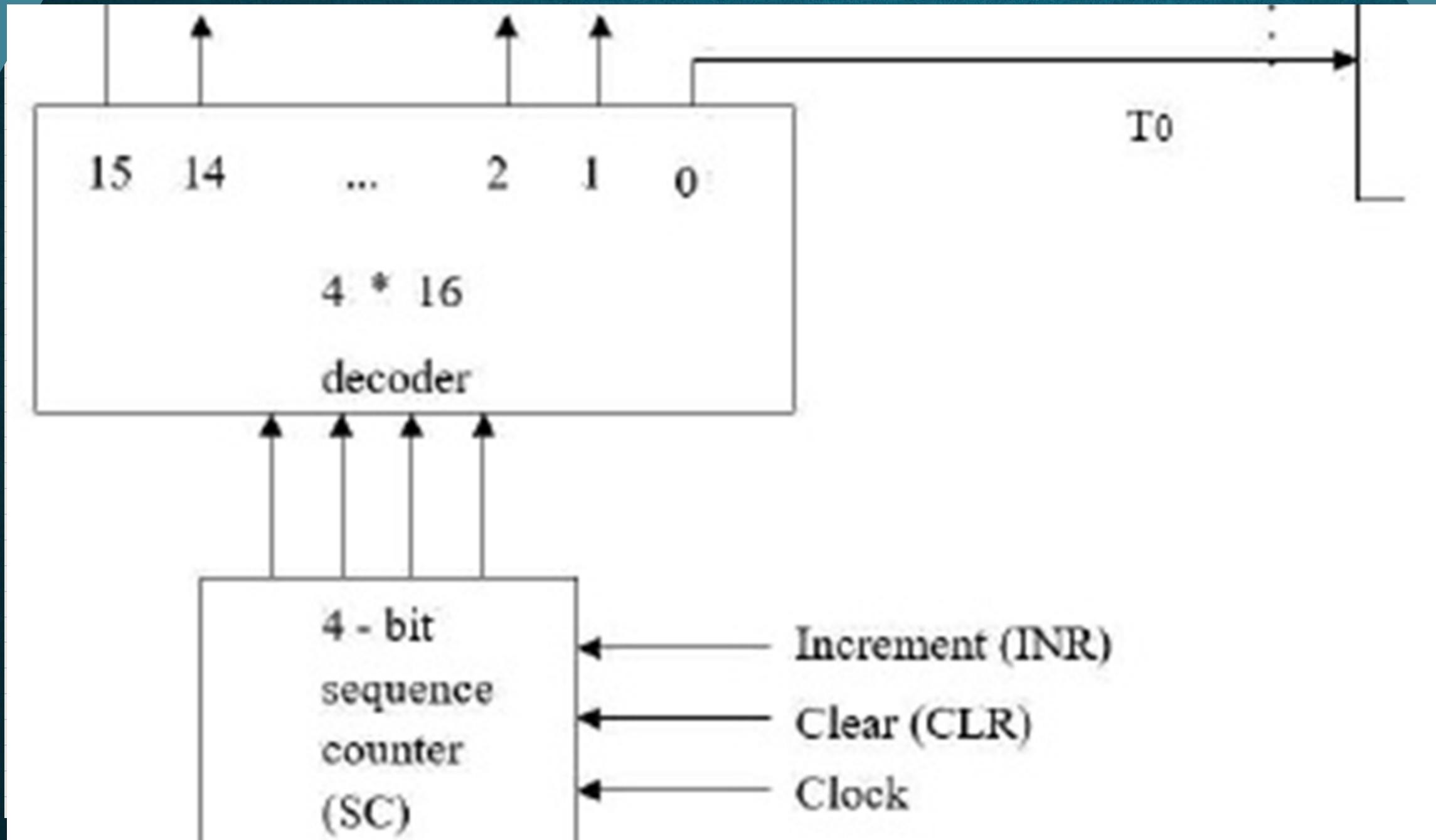


Control signals

```
-- Flip-Flops
signal I    : STD_LOGIC := '0';
signal E    : STD_LOGIC := '0';
signal R    : STD_LOGIC := '0';
signal IEN   : STD_LOGIC := '0';
signal FGI   : STD_LOGIC := '0';
signal FGO   : STD_LOGIC := '0';
```



COUNTER AND DECODER



COUNTER

```
begin
  process(clk)
  begin
    if rising_edge(clk) then
      if sc = '0' or count = "1111" then
        count <= (others => '0');
      else
        count <= std_logic_vector(unsigned(count) +
          1);
      end if;
    end if;
  end process;

  T <= count;
end Behavioral;
```



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity counter is
  Port (
    clk : in STD_LOGIC;
    sc  : in STD_LOGIC;
    T   : out STD_LOGIC_VECTOR(3 downto 0)
  );
end counter;

architecture Behavioral of counter is
  signal count : STD_LOGIC_VECTOR(3 downto 0) := "0000";
```



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Decoder4_16 is
    port(
        I      : in std_logic_vector(3 downto 0);
        Output: out std_logic_vector(15 downto 0)
    );
end Decoder4_16;

architecture arch_Decoder4_16 of Decoder4_16 is
begin
    process(I)
    begin
        Output <= (others => '0');

case I is
    when "0000" => Output <= "0000000000000001";
    when "0001" => Output <= "0000000000000010";
    when "0010" => Output <= "00000000000000100";
    when "0011" => Output <= "000000000000001000";
    when "0100" => Output <= "00000000000010000";
    when "0101" => Output <= "0000000000100000";
    when "0110" => Output <= "0000000001000000";
    when "0111" => Output <= "0000000010000000";
    when "1000" => Output <= "0000000100000000";
    when "1001" => Output <= "00000001000000000";
    when "1010" => Output <= "0000001000000000";
    when "1011" => Output <= "0000010000000000";
    when "1100" => Output <= "0001000000000000";
    when "1101" => Output <= "0010000000000000";
    when "1110" => Output <= "0100000000000000";
    when "1111" => Output <= "1000000000000000";
    when others => Output <= (others => '0');
end case;
end process;
end arch_Decoder4_16;
```



DECODER

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity Decoder4_16 is
    port(
        I      : in std_logic_vector(3 downto 0);
        Output: out std_logic_vector(15 downto 0)
    );
end Decoder4_16;

architecture arch_Decoder4_16 of Decoder4_16 is
begin
    process(I)
    begin
        Output <= (others => '0');

case I is
    when "0000" => Output <= "0000000000000001";
    when "0001" => Output <= "0000000000000010";
    when "0010" => Output <= "00000000000000100";
    when "0011" => Output <= "000000000000001000";
    when "0100" => Output <= "00000000000010000";
    when "0101" => Output <= "0000000000100000";
    when "0110" => Output <= "0000000001000000";
    when "0111" => Output <= "0000000010000000";
    when "1000" => Output <= "0000000100000000";
    when "1001" => Output <= "00000001000000000";
    when "1010" => Output <= "0000001000000000";
    when "1011" => Output <= "0000010000000000";
    when "1100" => Output <= "0001000000000000";
    when "1101" => Output <= "0010000000000000";
    when "1110" => Output <= "0100000000000000";
    when "1111" => Output <= "1000000000000000";
    when others => Output <= (others => '0');
end case;
end process;
end arch_Decoder4_16;
```



```

component counter is
  Port (
    clk : in STD_LOGIC;
    sc  : in STD_LOGIC;
    T   : out STD_LOGIC_VECTOR(3 downto 0)
  );
end component;

-- Decoder component declaration
component Decoder4_16 is
  port(
    I    : in std_logic_vector(3 downto 0);
    Output: out std_logic_vector(15 downto 0)
  );
end component;

-- Instantiate the 4-bit counter
Timing_Generator: counter
  port map(
    clk => clk,
    sc  => SC,
    T   => T_counter_4bit
  );
-- Instantiate the decoder
Decoder_inst: Decoder4_16
  port map(
    I => T_counter_4bit,
    Output => T
  );

```

/basic_computer/SC	1
/basic_computer/T_counter_4bit	4'h0
/basic_computer/T	16'b0000000000000000

```

-- Timing and Control
signal SC  : unsigned(3 downto 0) := "0000";
signal D   : STD_LOGIC_VECTOR(7 downto 0);

begin

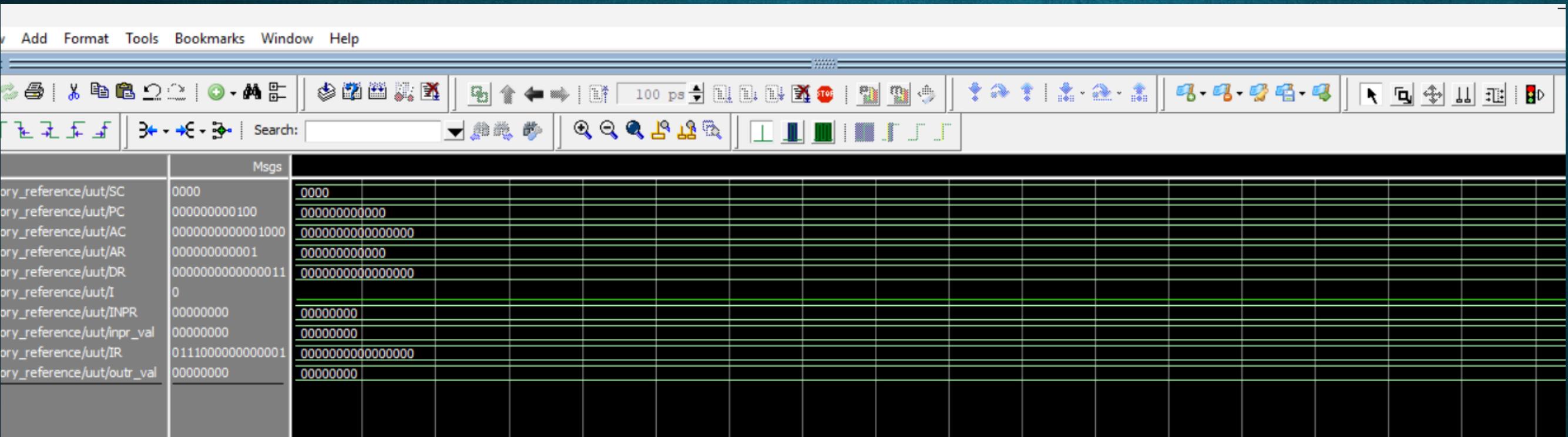
```

COUNTER AND DECODE
REPLACE
THIS WITH SIGNAL; SC



```
if reset = '1' then
    SC <= '0';
    PC <= "000000000010";
    AC <= (others => '0');
    AR <= (others => '0');
    IR <= (others => '0');
    DR <= (others => '0');
    TR <= (others => '0');
    OUTR_out <= (others => '0');
    OUTR_in <= (others => '0');
    INPR_out <= (others => '0');
    INPR_in <= (others => '0');
    temp <= (others => '0');
    I <= '0'; E <= '0'; R <= '0';
    IEN <= '0'; FGI <= '0'; FGO <= '0';
    D <= (others => '0');
    run <= true;
```

RESET



NORMAL INSTRUCTION CYC

```
elsif T(0) = '1' then
    AR <= PC;
    SC<='1';
elsif T(1) = '1' then
    IR <= RAM(to_integer(unsigned(AR)));
    PC <= std_logic_vector(unsigned(PC) + 1);
    SC<='1';
elsif T(2) = '1' then
    D <= IR(14 downto 12);
    AR <= IR(11 downto 0);
    I <= IR(15);
    SC<='1';
```

+--> /tb_memory_reference/uut/SC	0000	0000	0001	0010	0011	0100	0101	0110	0000
+--> /tb_memory_reference/uut/PC	000000000100	000000000000		000000000001					
+--> /tb_memory_reference/uut/AC	00000000000001000	0000000000000000						000000000000101	
+--> /tb_memory_reference/uut/AR	00000000000001	00000000000000			0000000001010				
+--> /tb_memory_reference/uut/DR	00000000000000011	0000000000000000				000000000101			
+--> /tb_memory_reference/uut/I	0	0							
+--> /tb_memory_reference/uut/INPR	00000000	00000000							
+--> /tb_memory_reference/uut/inpr_val	00000000	00000000							
+--> /tb_memory_reference/uut/IR	0111000000000001	0000000000000000	0010000000001010						

AND

```
when "0101" =>
    if D(0) = '1' then AC <= AC and DR; -- AND
```

ADD

```
if D(1) = '1' then -- ADD
temp_add := unsigned('0' & AC) + unsigned('0' & DR);
AC <= std_logic_vector(temp_add(15 downto 0));
E <= temp_add(16);
```



Memory reference instructions

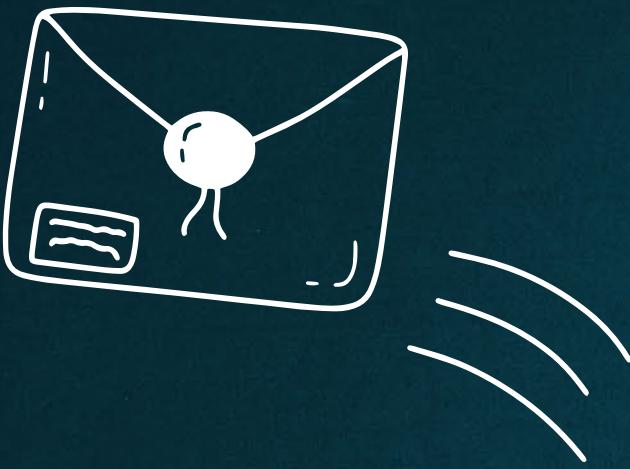
STA



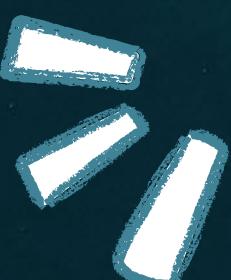
LDA

```
elsif D(1) = '1' then AC <= DR; -- LDA
else if D(5) = '1' then PC <= DR; -- PC
```

```
if D(0)='1' or D(1)='1' or D(2)='1' or D(6)='1' then
    DR <= RAM(to_integer(unsigned(AR)));
elsif D(3) = '1' then -- STA
    RAM(to_integer(unsigned(AR))) <= AC;
    SC <= "0000";
```



	Msgs	Data											
iut/SC	0000	0000	0001	0010	0011	0100	0101	0110	0000	0001	0010	0011	0100
iut/PC	000...	000000000000	000000000001								000000000010		
iut/AC	000...	0000000000000000						000000000000101					
iut/AR	000...	000000000000		000000001010				000000000000...	00000000...	000000001011			
iut/DR	000...	0000000000000000				000000000000101							
iut/I	0						000000000000101						
iut/INPR	000...	00000000											
iut/inpr_val	000...	00000000											
iut/IR	011...	0000000000000000	0010000000001010						0001000000001011				



MEMORY REFERENCE INSTRUCTION

BUN

```
when "100" => --- BUN  
    PC <= AR;  
    SC<='0';
```



+/- /basic_computer/PC	000000000110	00000001000			000000001001		000000111111	
+/- /basic_computer/AR	100000000000	00000... 010000000000			000000001000		000000111111	
+/- /basic_computer/IR	0111100000000000	0111010000000000			0100000000111111			
+/- /basic_computer/DR	000000010101010	000000010101010						
+/- /basic_computer/AC	111111101010101	000000000000000						
+/- /basic_computer/O...	00000000	00000000						
+/- /basic_computer/O...	00000000	00000000						
+/- /basic_computer/IN...	00000000	00000000						
+/- /basic_computer/IN...	00000000	00000000						
+/- /basic_computer/temp	0000000011111...	000000001111111						
+/- /basic_computer/TR	0000000000000000	0000000000000000						

MEMORY REFERENCE INSTRUCTI

BSA

```
when "101" => -- BSA
    RAM(to_integer(unsigned(AR)) ) <= "0000" & PC ;
    AR <= std_logic_vector(unsigned(AR) + 1);
    SC<='1';

    PC <= AR;
    SC<='0';
```

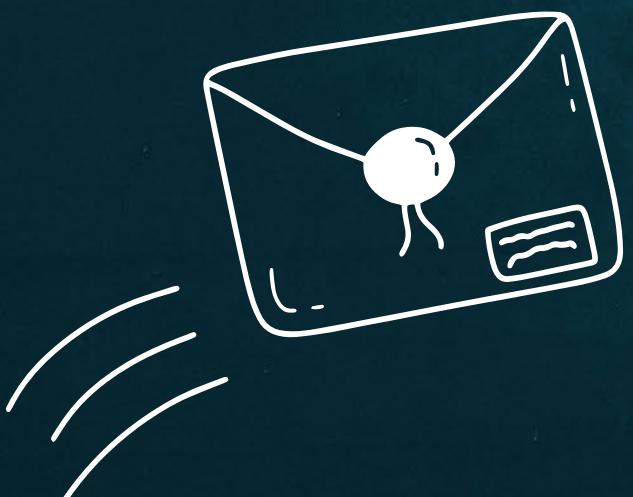
/basic_computer/PC	00000001001	00000... 0000011111	00001000000	000100000001
/basic_computer/AR	000000111111	000000111111	000100000000	000100000001
/basic_computer/IR	0100000000111111	0100000000111111	0101000100000000	
/basic_computer/DR	0000000010101010	0000000010101010		
/basic_computer/AC	0000000000000000	0000000000000000		
/basic_computer/O...	00000000	00000000		
/basic_computer/O...	00000000	00000000		
/basic_computer/IN...	00000000	00000000		
/basic_computer/IN...	00000000	00000000		
/basic_computer/temp	00000000011111...	0000000001111111		
/basic_computer/TR	0000000000000000	0000000000000000		

MEMORY REFERENCE INSTRUCTION

ISZ

```
elsif D(6) = '1' then -- ISZ
    DR <= std_logic_vector(unsigned(DR) + 1);
end if;

-- T6: Finalize ISZ
when "0110" =>
    if D(6) = '1' then
        RAM(to_integer(unsigned(AR))) <= DR;
        if DR = x"0000" then
            PC <= std_logic_vector(unsigned(PC) + 1);
        end if;
    end if;
    SC <= "0000";
```



/basic_computer/PC	000001000000	000001000000	000100000001	000100000010	000100000011
/basic_computer/AR	000100000001	000100...000100000001	000100000001	000100110000	
/basic_computer/IR	0101000100000000	0101000100000000		0110000100110000	
/basic_computer/DR	0000000010101010	0000000010101010			1111111111111111
/basic_computer/AC	0000000000000000	0000000000000000			0000000000000000
/basic_computer/O...	00000000	00000000			
/basic_computer/O...	00000000	00000000			
/basic_computer/IN...	00000000	00000000			
/basic_computer/IN...	00000000	00000000			
/basic_computer/temp	0000000011111111	0000000011111111			
/basic_computer/TR	0000000000000000	0000000000000000			

REGISTER REFERENCE

CLEAR AC



```
-- Register Reference and I/O Instructions
elsif T(3) = '1' then
    if I = '0' and D = "111" then
        -- Register reference instructions
        if IR(11) = '1' then
            AC <= (others => '0'); -- CLA
```

COMPLEMENT AC

```
        ...  
    elsif IR(9) = '1' then  
        AC <= not AC; -- CMA  
    end if;
```


| | Msgs | ter/dk | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 | 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 | 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 | 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 | 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 | 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | 257 | 258 | 259 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | 267 | 268 | 269 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | 277 | 278 | 279 | 280 | 281 | 282 | 283 | 284 | 285 | 286 | 287 | 288 | 289 | 290 | 291 | 292 | 293 | 294 | 295 | 296 | 297 | 298 | 299 | 300 | 301 | 302 | 303 | 304 | 305 | 306 | 307 | 308 | 309 | 310 | 311 | 312 | 313 | 314 | 315 | 316 | 317 | 318 | 319 | 320 | 321 | 322 | 323 | 324 | 325 | 326 | 327 | 328 | 329 | 330 | 331 | 332 | 333 | 334 | 335 | 336 | 337 | 338 | 339 | 340 | 341 | 342 | 343 | 344 | 345 | 346 | 347 | 348 | 349 | 350 | 351 | 352 | 353 | 354 | 355 | 356 | 357 | 358 | 359 | 360 | 361 | 362 | 363 | 364 | 365 | 366 | 367 | 368 | 369 | 370 | 371 | 372 | 373 | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 | 382 | 383 | 384 | 385 | 386 | 387 | 388 | 389 | 390 | 391 | 392 | 393 | 394 | 395 | 396 | 397 | 398 | 399 | 400 | 401 | 402 | 403 | 404 | 405 | 406 | 407 | 408 | 409 | 410 | 411 | 412 | 413 | 414 | 415 | 416 | 417 | 418 | 419 | 420 | 421 | 422 | 423 | 424 | 425 | 426 | 427 | 428 | 429 | 430 | 431 | 432 | 433 | 434 | 435 | 436 | 437 | 438 | 439 | 440 | 441 | 442 | 443 | 444 | 445 | 446 | 447 | 448 | 449 | 450 | 451 | 452 | 453 | 454 | 455 | 456 | 457 | 458 | 459 | 460 | 461 | 462 | 463 | 464 | 465 | 466 | 467 | 468 | 469 | 470 | 471 | 472 | 473 | 474 | 475 | 476 | 477 | 478 | 479 | 480 | 481 | 482 | 483 | 484 | 485 | 486 | 487 | 488 | 489 | 490 | 491 | 492 | 493 | 494 | 495 | 496 | 497 | 498 | 499 | 500 | 501 | 502 | 503 | 504 | 505 | 506 | 507 | 508 | 509 | 510 | 511 | 512 | 513 | 514 | 515 | 516 | 517 | 518 | 519 | 520 | 521 | 522 | 523 | 524 | 525 | 526 | 527 | 528 | 529 | 530 | 531 | 532 | 533 | 534 | 535 | 536 | 537 | 538 | 539 | 540 | 541 | 542 | 543 | 544 | 545 | 546 | 547 | 548 | 549 | 550 | 551 | 552 | 553 | 554 | 555 | 556 | 557 | 558 | 559 | 560 | 561 | 562 | 563 | 564 | 565 | 566 | 567 | 568 | 569 | 570 | 571 | 572 | 573 | 574 | 575 | 576 | 577 | 578 | 579 | 580 | 581 | 582 | 583 | 584 | 585 | 586 | 587 | 588 | 589 | 590 | 591 | 592 | 593 | 594 | 595 | 596 | 597 | 598 | 599 | 600 | 601 | 602 | 603 | 604 | 605 | 606 | 607 | 608 | 609 | 610 | 611 | 612 | 613 | 614 | 615 | 616 | 617 | 618 | 619 | 620 | 621 | 622 | 623 | 624 | 625 | 626 | 627 | 628 | 629 | 630 | 631 | 632 | 633 | 634 | 635 | 636 | 637 | 638 | 639 | 640 | 641 | 642 | 643 | 644 | 645 | 646 | 647 | 648 | 649 | 650 | 651 | 652 | 653 | 654 | 655 | 656 | 657 | 658 | 659 | 660 | 661 | 662 | 663 | 664 | 665 | 666 | 667 | 668 | 669 | 670 | 671 | 672 | 673 | 674 | 675 | 676 | 677 | 678 | 679 | 680 | 681 | 682 | 683 | 684 | 685 | 686 | 687 | 688 | 689 | 690 | 691 | 692 | 693 | 694 | 695 | 696 | 697 | 698 | 699 | 700 | 701 | 702 | 703 | 704 | 705 | 706 | 707 | 708 | 709 | 710 | 711 | 712 | 713 | 714 | 715 | 716 | 717 | 718 | 719 | 720 | 721 | 722 | 723 | 724 | 725 | 726 | 727 | 728 | 729 | 730 | 731 | 732 | 733 | 734 | 735 | 736 | 737 | 738 | 739 | 740 | 741 | 742 | 743 | 744 | 745 | 746 | 747 | 748 | 749 | 750 | 751 | 752 | 753 | 754 | 755 | 756 | 757 | 758 | 759 | 760 | 761 | 762 | 763 | 764 | 765 | 766 | 767 | 768 | 769 | 770 | 771 | 772 | 773 | 774 | 775 | 776 | 777 | 778 | 779 | 780 | 781 | 782 | 783 | 784 | 785 | 786 | 787 | 788 | 789 | 790 | 791 | 792 | 793 | 794 | 795 | 796 | 797 | 798 | 799 | 800 | 801 | 802 | 803 | 804 | 805 | 806 | 807 | 808 | 809 | 810 | 811 | 812 | 813 | 814 | 815 | 816 | 817 | 818 | 819 | 820 | 821 | 822 | 823 | 824 | 825 | 826 | 827 | 828 | 829 | 830 | 831 | 832 | 833 | 834 | 835 | 836 | 837 | 838 | 839 | 840 | 841 | 842 | 843 | 844 | 845 | 846 | 847 | 848 | 849 | 850 | 851 | 852 | 853 | 854 | 855 | 856 | 857 | 858 | 859 | 860 | 861 | 862 | 863 | 864 | 865 | 866 | 867 | 868 | 869 | 870 | 871 | 872 | 873 | 874 | 875 | 876 | 877 | 878 | 879 | 880 | 881 | 882 | 883 | 884 | 885 | 886 | 887 | 888 | 889 | 890 | 891 | 892 | 893 | 894 | 895 | 896 | 897 | 898 | 899 | 900 | 901 | 902 | 903 | 904 | 905 | 906 | 907 | 908 | 909 | 910 | 911 | 912 | 913 | 914 | 915 | 916 | 917 | 918 | 919 | 920 | 921 | 922 | 923 | 924 | 925 | 926 | 927 | 928 | 929 | 930 | 931 | 932 | 933 | 934 | 935 | 936 | 937 | 938 | 939 | 940 | 941 | 942 | 943 | 944 | 945 | 946 | 947 | 948 |<
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

COMPLEMENT E

```
elsif IR(8) = '1' then  
    E <= not E; -- CME
```



	Msgs
/basic_computer/dk	-No Data-
/basic_computer/reset	-No Data-
/basic_computer/PC	-No Data-
/basic_computer/AR	-No Data-
/basic_computer/IR	-No Data-
/basic_computer/DR	-No Data-
/basic_computer/AC	-No Data-
/basic_computer/O...	-No Data-
/basic_computer/O...	-No Data-
/basic_computer/IN...	-No Data-
/basic_computer/IN...	-No Data-
/basic_computer/temp	-No Data-
/basic_computer/TR	-No Data-
/basic_computer/I	-No Data-
/basic_computer/E	-No Data-
/basic_computer/R	-No Data-

CLEAR E

```
elsif IR(10) = '1' then  
    E <= '0'; -- CLE
```



	Msgs
computer/dk	0
computer/reset	0
computer/PC	12'h007
computer/AR	12'h400
computer/IR	16'h7400
computer/DR	16'h00AA
computer/AC	16'hFFFF
computer/O...	8'h00
computer/O...	8'h00
computer/IN...	8'h00
computer/IN...	8'h00
computer/temp	17h00000
computer/TR	16'h0000
computer/I	0
computer/E	0
computer/R	0
computer/EN	0

CIRCULATE RIGHT AC

```

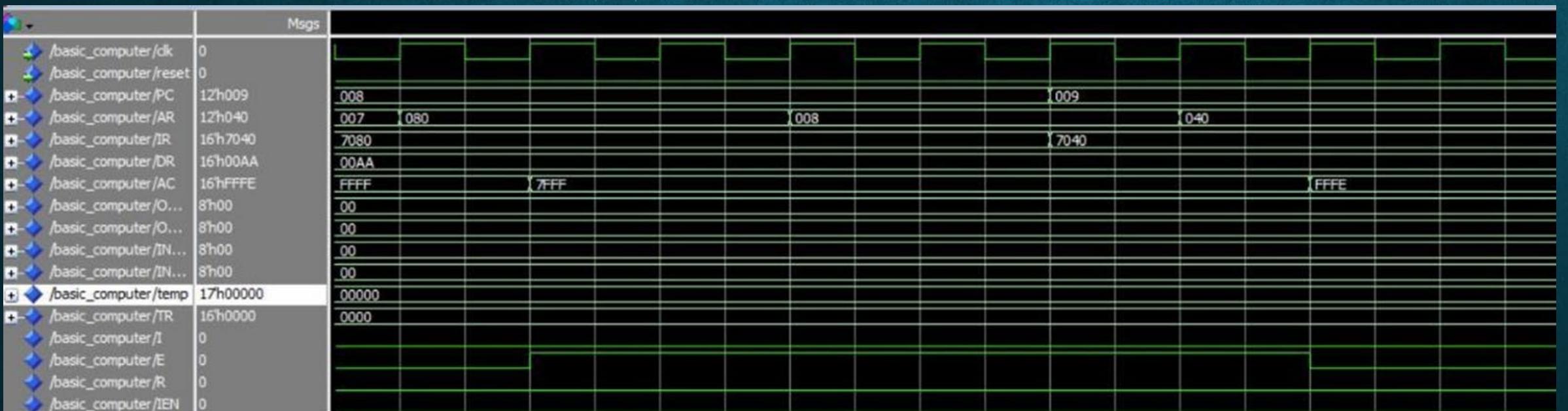
if IR(7) = '1' then -- SHR
    E <= AC(0); AC <= E & AC(15 downto 1);
end if;

```



CIRCULATE LEFT AC

```
E <= AC(0); AC <= E & AC(15 downto 1);  
end if;  
if IR(6) = '1' then -- SHL  
    E <= AC(15); AC <= AC(14 downto 0) & E;  
end if;
```



INCREMENT AC

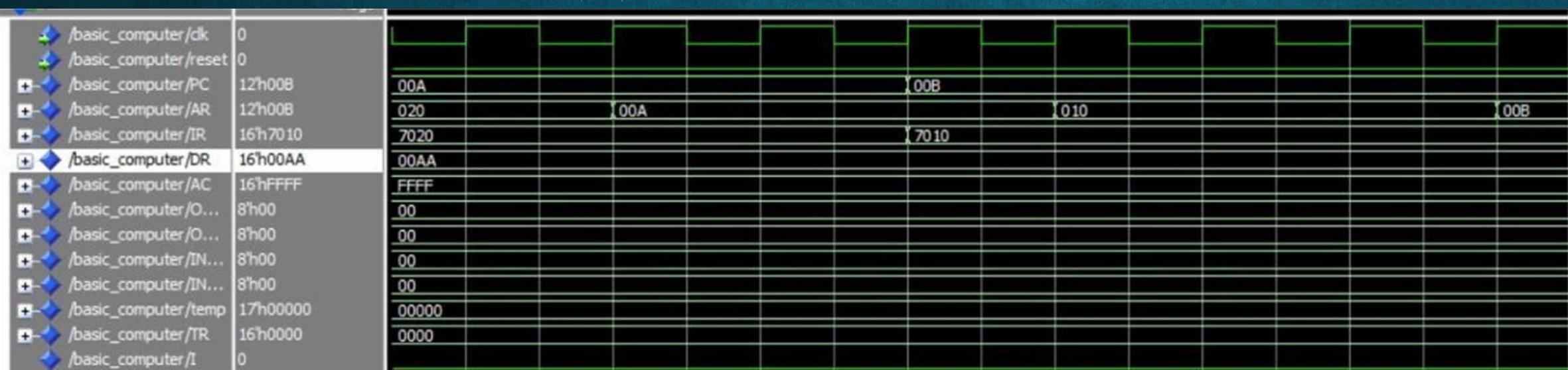
```
elsif IR(5) = '1' then
    AC <= std_logic_vector(unsigned(AC) + 1); -- INC
```

	/basic_computer/dk	0	/basic_computer/reset	0	
+-->	/basic_computer/PC	12h00A			
+-->	/basic_computer/AR	12h020			
+-->	/basic_computer/IR	16h7020			
+-->	/basic_computer/DR	16h00AA			
+-->	/basic_computer/AC	16hFFFF			
+-->	/basic_computer/O...	8'h00			
+-->	/basic_computer/O...	8'h00			
+-->	/basic_computer/IN...	8'h00			
+-->	/basic_computer/IN...	8'h00			
+-->	/basic_computer/temp	17h00000			
+-->	/basic_computer/TR	16h0000			
+-->	/basic_computer/I	0			
+-->	/basic_computer/E	0			
+-->	/basic_computer/R	0			



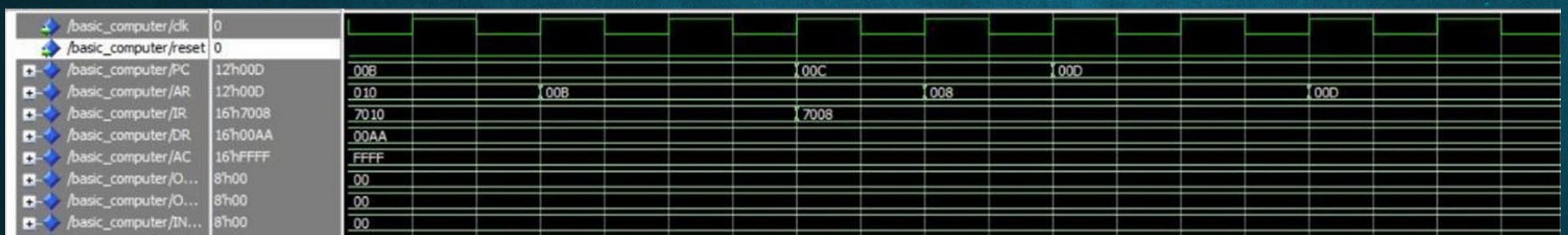
SKIP IF AC (+)

```
elsif IR(4) = '1' then
    if AC(15) = '0' and AC /= x"0000" then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SPA
    end if;
```



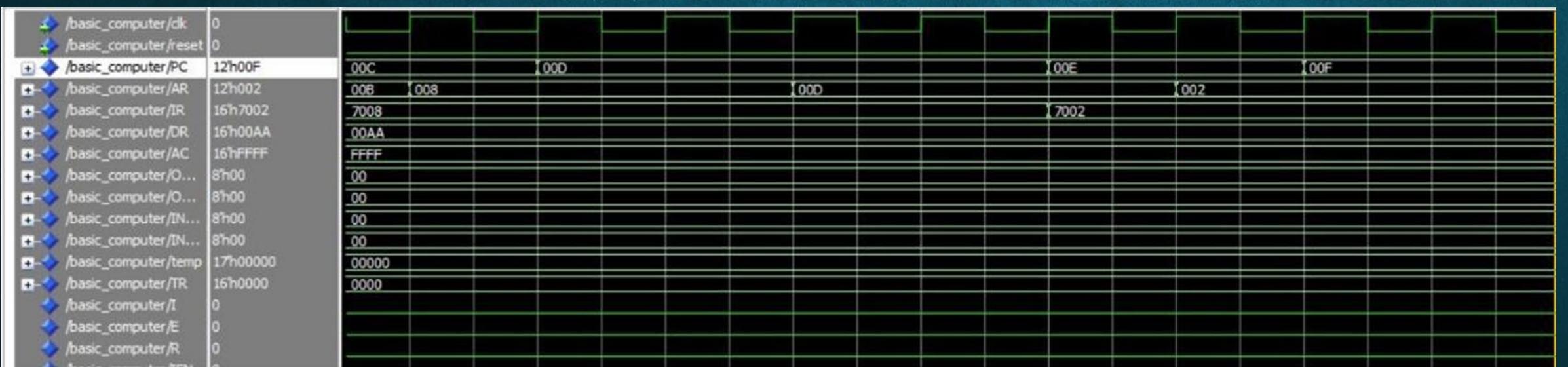
SKIP IF AC >

```
elsif IR(3) = '1' then
    if AC(15) = '1' then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SNA
    end if;
```



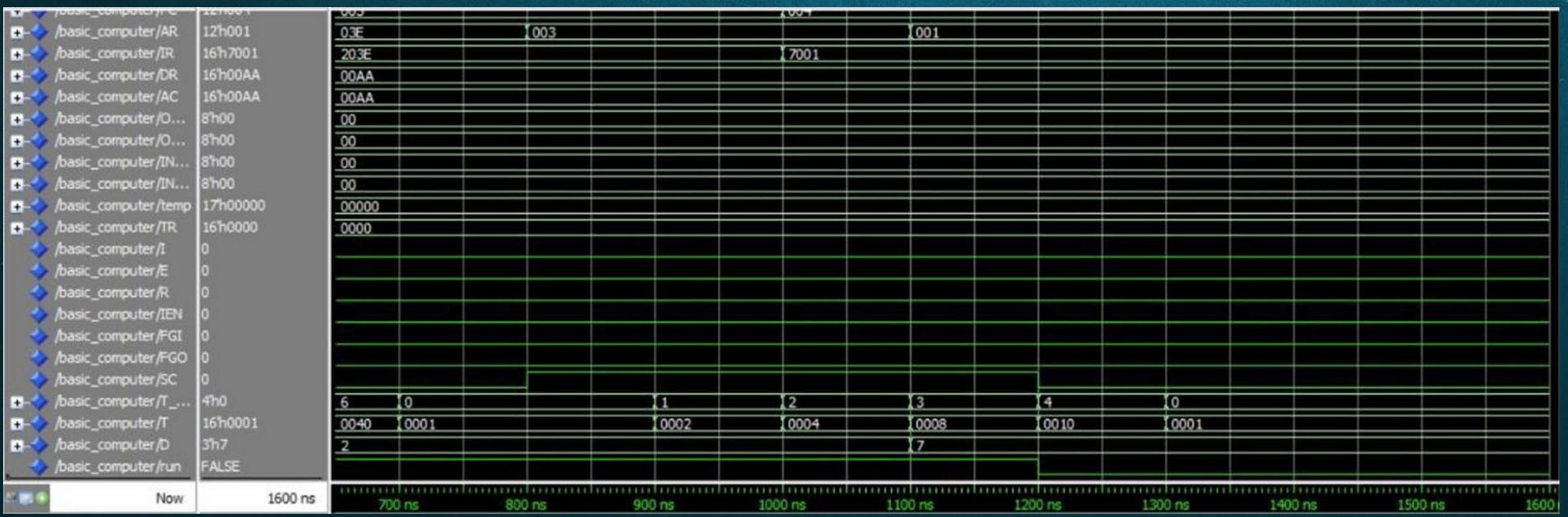
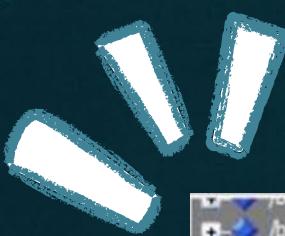
SKIP IF E=0

```
elsif IR(1) = '1' then
    if E = '0' then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SZE
    end if;
```



HLT

```
elsif IR(0) = '1' then  
    run <= false; -- HLT  
end if;
```



REGISTER REFERE INSTRUCTIONS

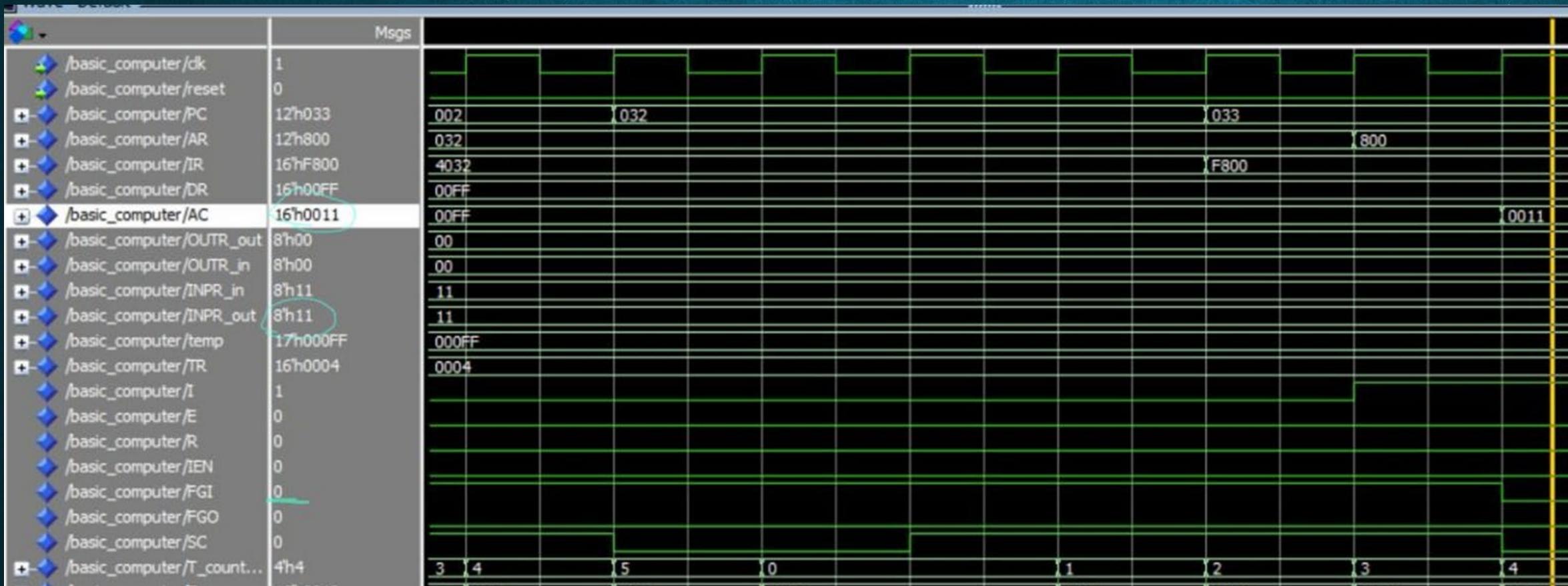
```
-- Register Reference and I/O Instructions
elsif T(3) = '1' then
    if I = '0' and D = "111" then
        -- Register reference instructions
        if IR(11) = '1' then
            AC <= (others => '0'); -- CLA
        elsif IR(10) = '1' then
            E <= '0'; -- CLE
        elsif IR(9) = '1' then
            AC <= not AC; -- CMA
        elsif IR(8) = '1' then
            E <= not E; -- CME
        elsif IR(5) = '1' then
            AC <= std_logic_vector(unsigned(AC) + 1); -- INC
        elsif IR(6) = '1' then
            E <= AC(15);
            AC <= AC(14 downto 0) & '0'; -- SHL
        elsif IR(7) = '1' then
            E <= AC(0);
            AC <= '0' & AC(15 downto 1); -- SHR
```

```
elsif IR(4) = '1' then
    if AC(15) = '0' and AC /= x"0000" then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SPA
    end if;
elsif IR(3) = '1' then
    if AC(15) = '1' then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SNA
    end if;
elsif IR(2) = '1' then
    if AC = x"0000" then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SZA
    end if;
elsif IR(1) = '1' then
    if E = '0' then
        PC <= std_logic_vector(unsigned(PC) + 1); -- SZE
    end if;
elsif IR(0) = '1' then
    run <= false; -- HLT
end if;
```

INPUT OUTPUT INSTRUCTIONS

1)
INP

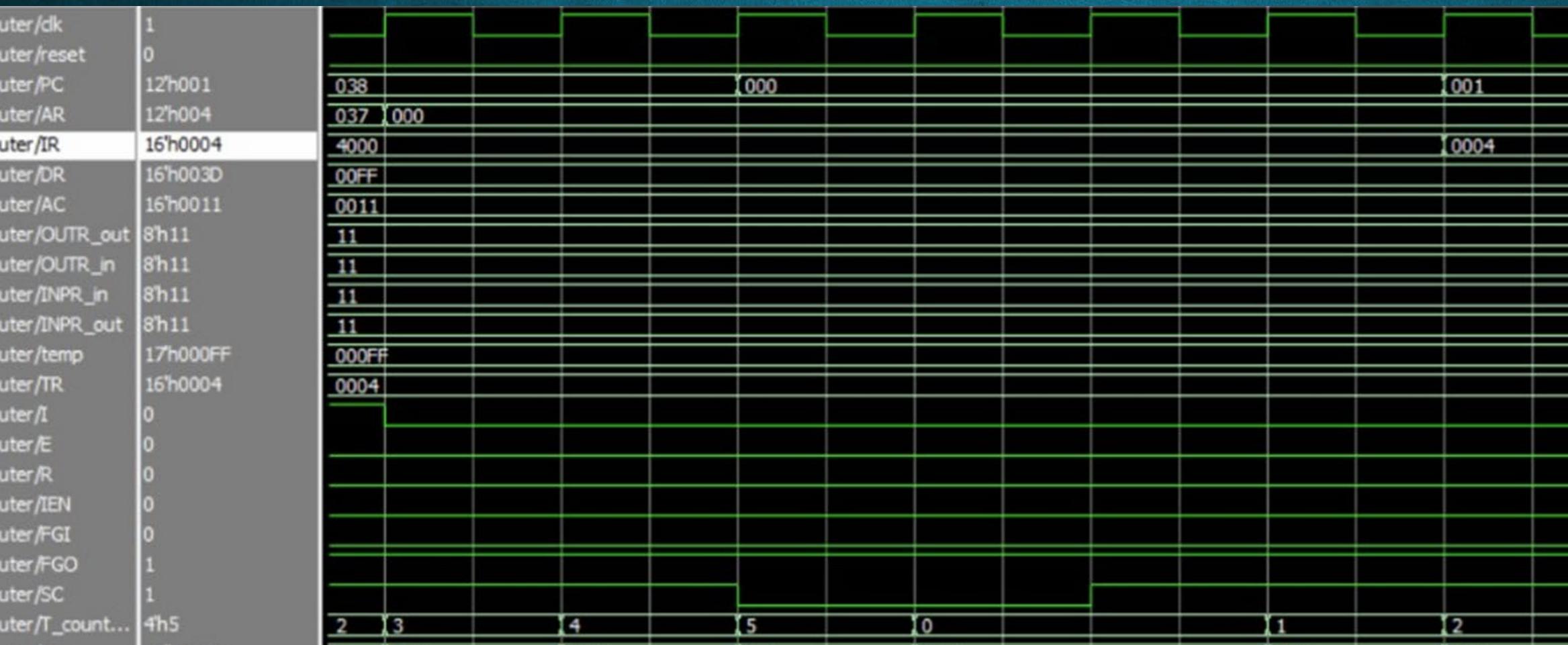
30	INP	F800	AC(7:0) ← INPR; FGI ← 0	T3	AC lower 8 bits updated, FGI cleared
--- I/O instructions if IR(11) = '1' then AC <= AC(15 downto 8) & INPR_out; FGI <= '0'; -- INP					



INPUT OUTPUT INSTRUCTIONS

2)
OUT

31	OUT	F400	OUTR \leftarrow AC(7:0); FGO \leftarrow 0	T3	OUTR updated, FGO cleared
elsif IR(10) = '1' then OUTR_in <= AC(7 downto 0); FGO <= '0'; -- OUT					

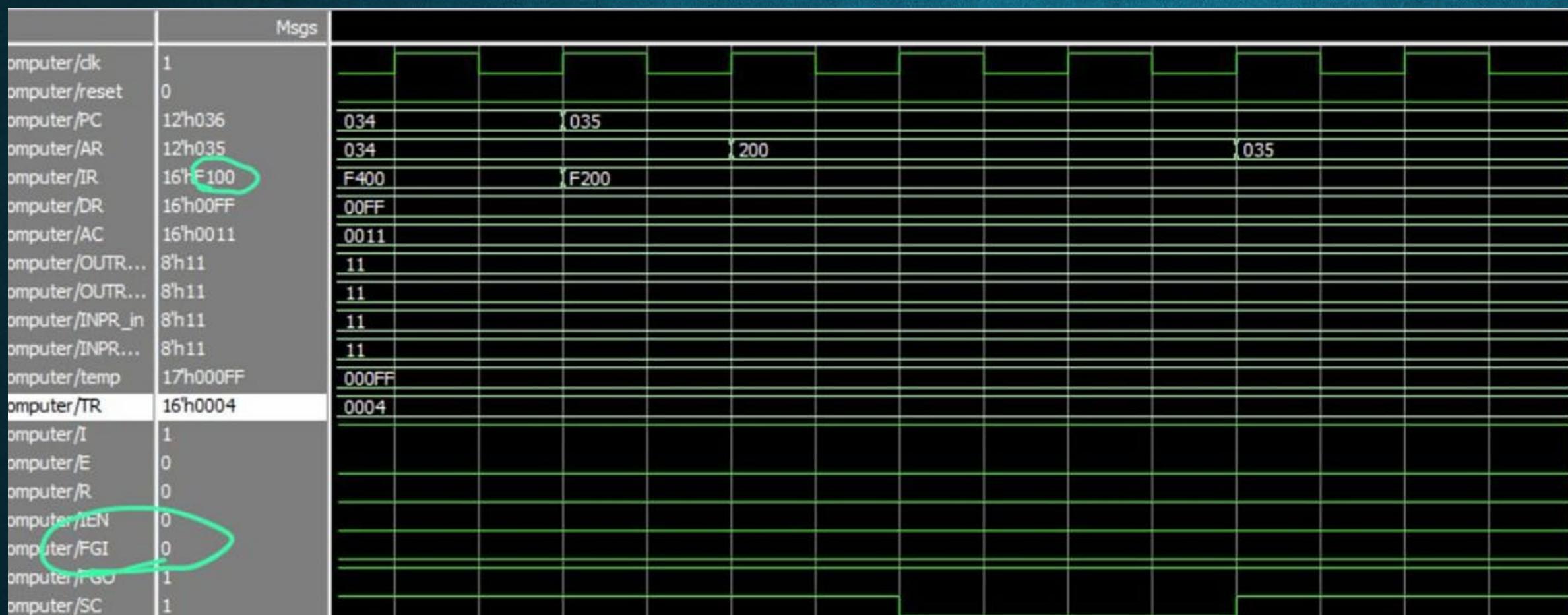


INPUT OUTPUT INSTRUCTIONS

3)

SKI

32	SKI	F200	if FGI=1, PC++	T3	PC incremented if FGI=1
<pre>if IR(9) = '1' and FGI = '1' then PC <= std_logic_vector(unsigned(PC)+1); end if; if IR(8) = '1' and FGO = '1' then PC <= std_logic_vector(unsigned(PC)+1); end if;</pre>					



IR9 =1
BUT FGI NOT EQUAL 1 NO SKI

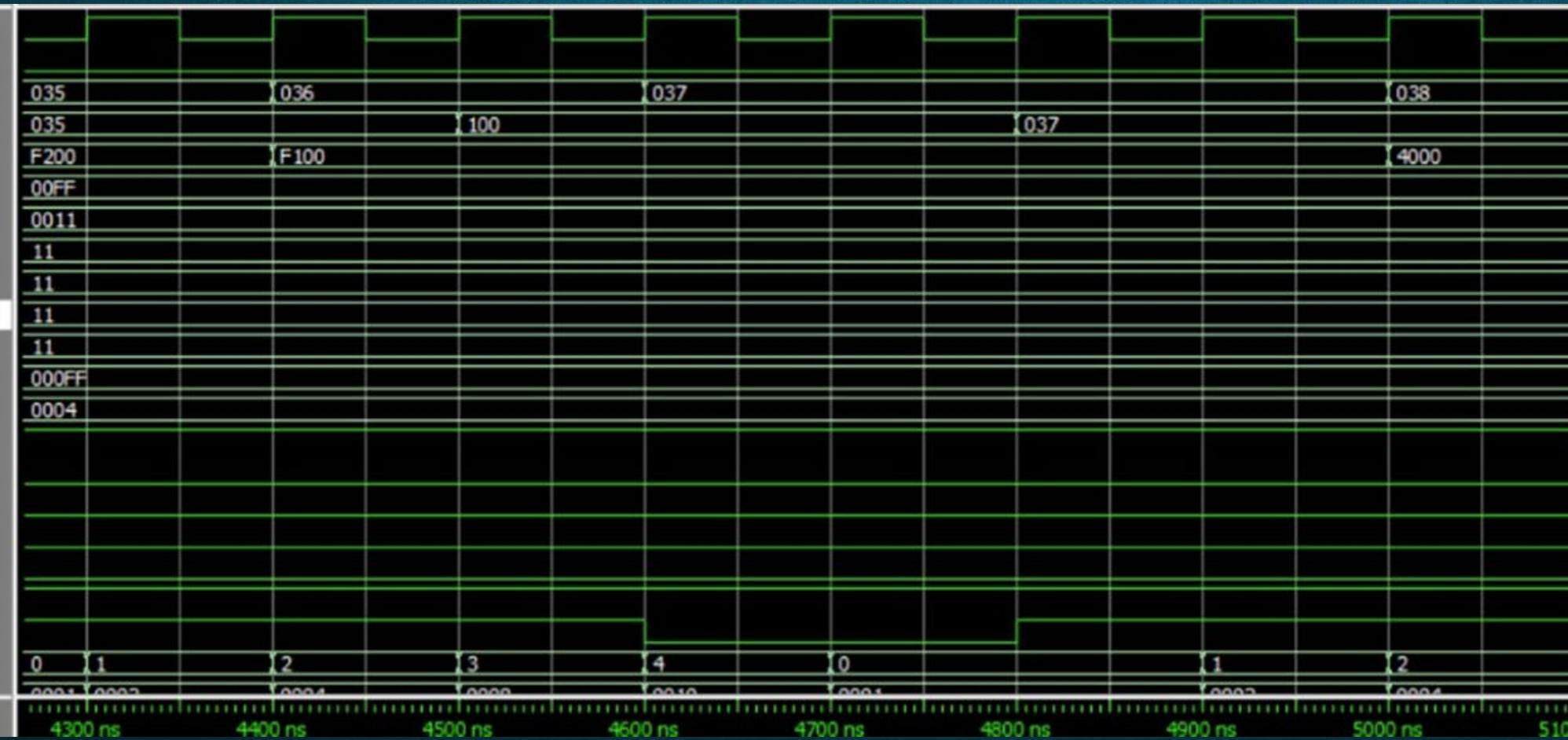
INPUT OUTPUT INSTRUCTIONS

4)

SKO

33	SKO	F100	if FGO=1, PC++	T3	PC incremented if FGO=1
<pre>if IR(9) = '1' and FGI = '1' then PC <= std_logic_vector(unsigned(PC)+1); end if; if IR(8) = '1' and FGO = '1' then PC <= std_logic_vector(unsigned(PC)+1); end if;</pre>					

ter/dk	-No Data-
ter/reset	-No Data-
ter/PC	-No Data-
ter/AR	-No Data-
ter/IR	-No Data-
ter/DR	-No Data-
ter/AC	-No Data-
ter/OUTR...	-No Data-
ter/OUTR...	-No Data-
ter/INPR_in	-No Data-
ter/INPR...	-No Data-
ter/temp	-No Data-
ter/TR	-No Data-
ter/I	-No Data-
ter/E	-No Data-
ter/R	-No Data-
ter/IEN	-No Data-
ter/FGI	-No Data-
ter/FGO	-No Data-
ter/SC	-No Data-
ter/T_cou...	-No Data-
ter/T	-No Data-



IR 8=1
FGO=1
SO PC=PC+1=037



INPUT OUTPUT INSTRUCTIONS

5)
I -> ON

```
elsif IR(7) = '1' then
    IEN <= '1'; -- ION
```



6)

I->OF

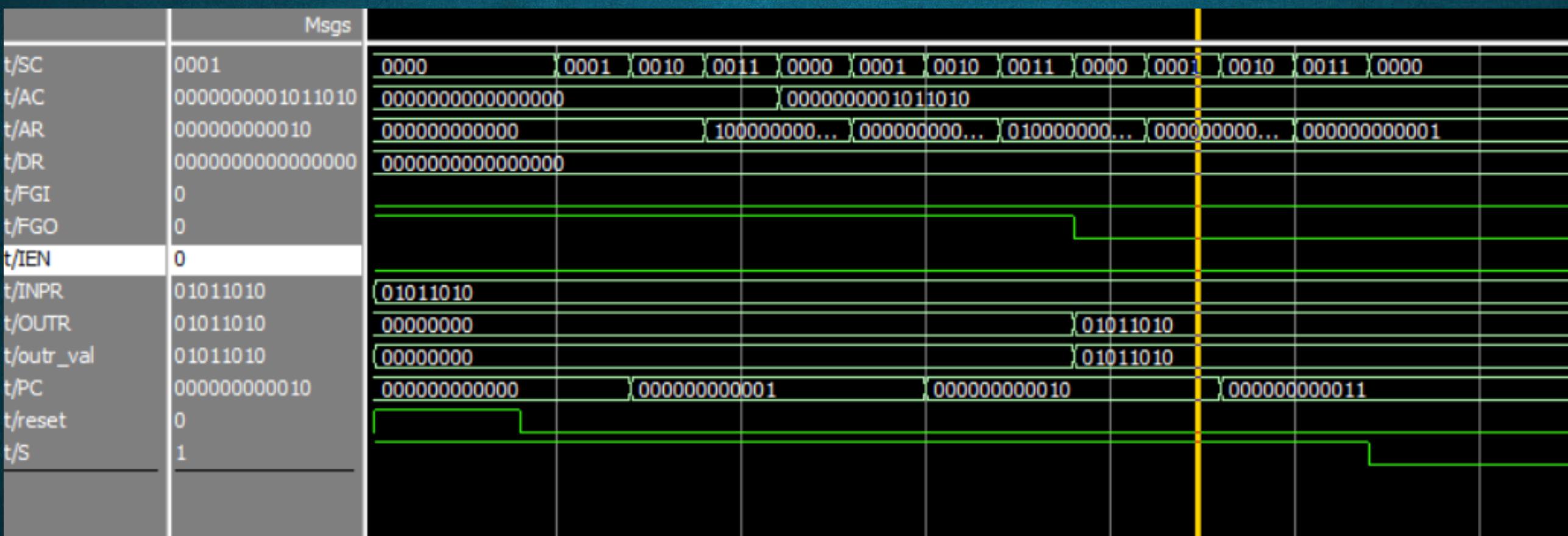
INPUT OUTPUT INSTRUCTIONS

IOF

```
      pB6:    IEN ← 0  
      elseif IR(6) = '1' then  
          IEN <= '0'; -- IOF  
      end if;
```



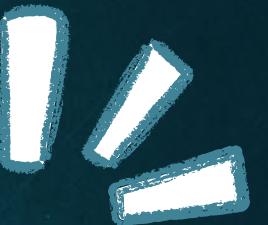
INTERRUPT AND I/O FLAG HANDLING LOGIC



INTERRUPT AND I/O FLAG HANDLING LOGIC

Interrupt Check

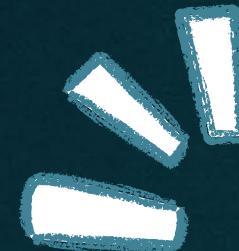
```
if IR(7) = '1' then IEN <= '1'; end if; -- ION  
if IR(6) = '1' then IEN <= '0'; end if; -- IOF
```



ut/FGO	1
ut/I	1
ut/IEN	0
ut/INPR	01011010
ut/IR	1111100000000000
ut/PC	000000000001
ut/SC	0001



THANK YOU



ESLAM AHMED



MOHAMED EMAD



BASIL MAGDY