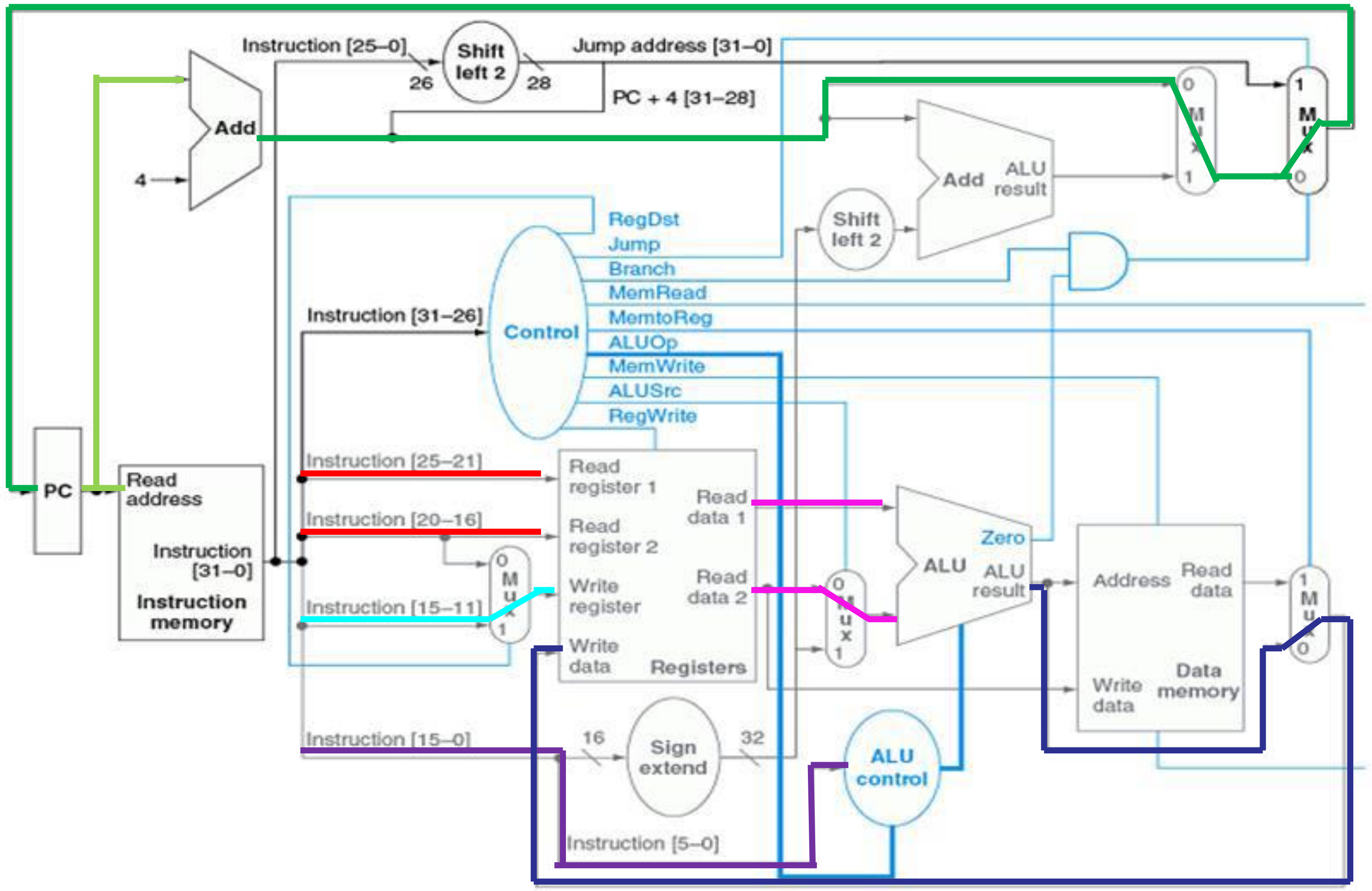
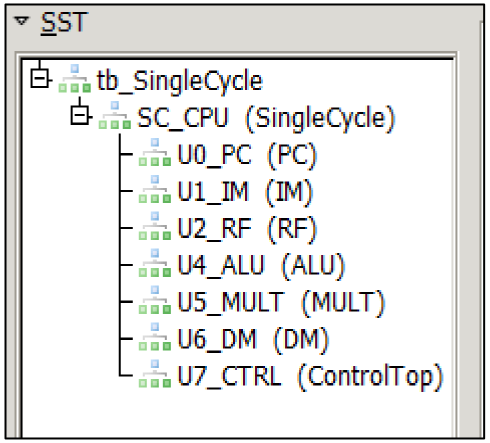
Computer Architecture Lab

04. Week #5

Single Cycle CPU_ex.subu



Single Cycle CPU module



```
SST
tb_SingleCycle
  SC_CPU  (SingleCycle)
    U0_PC  (PC)
    U1_IM  (IM)
    U2_RF  (RF)
    U4_ALU  (ALU)
    U5_MULT  (MULT)
    U6_DM  (DM)
    U7_CTRL  (ControlTop)
```

Table 1 – Instance name of top module

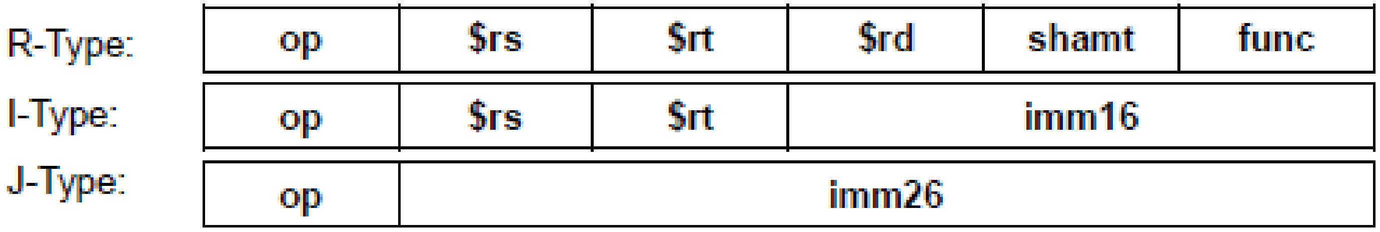
Instance name	Description
SC_CPU	Top module. Single-Cycle CPU
U0_PC	Program counter
U1_IM	Instruction memory
U2_RF	Register file
U3_SEU	Sign Extension Unit
U4_ALU	Arithmetic Logic Unit
U5_MULT	Multiplier Logic Unit
U6_DM	Data memory
U7_CTRL	Control unit -MainControl, MyControl

PLA_AND.txt

Table 2 – Consecutive Instruction Decoding Configuration in PLA_AND.txt

Port name	Classification	Bit	Description
Op	Input	6-bit	Op code
Func	Input	6-bit	Function code
Regimm	Input	5-bit	Register Immediate code (RT)

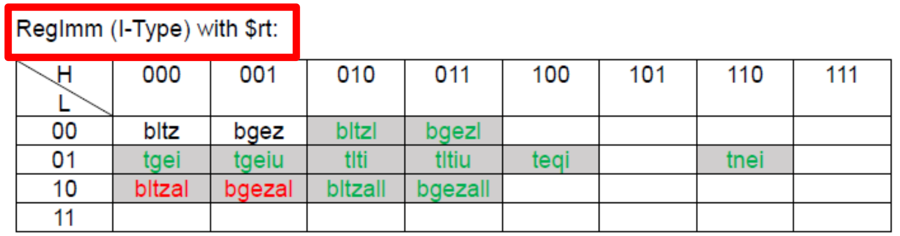
MIPS Inst Datasheet.pdf



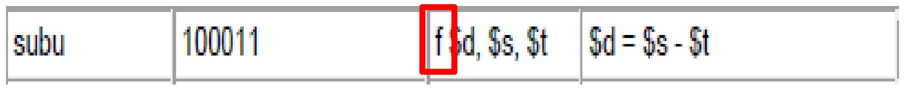
R-Type:	op	$rs	$rt	$rd	shamt	func
I-Type:	op	$rs	$rt	imm16		
J-Type:	op	imm26				

PLA_AND.txt

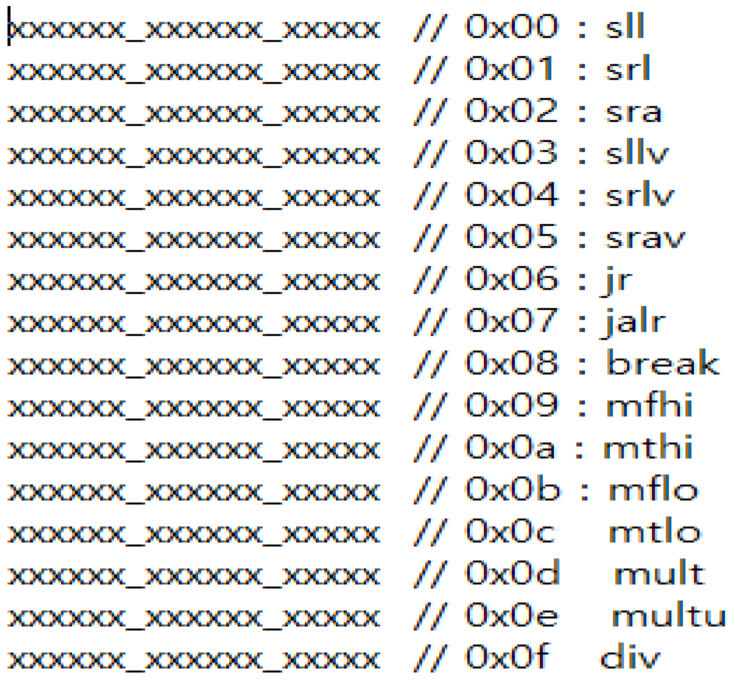
- R-type's opcode is 000000
- When implementing R-type instructions, you should fill Function
- RegImm is filled when you implement specific I-type instructions (ex.bltz, bgez)


RegImm (I-Type) with $rt:

H / L	000	001	010	011	100	101	110	111
00	bltz	bgez	bltzl	bgezl				
01	tgei	tgeiu	tlti	tltiu	teqi		tnei	
10	bltzal	bgezal	bltzall	bgezall				
11								



subu	100011	f $d, $s, $t	$d = $s - $t

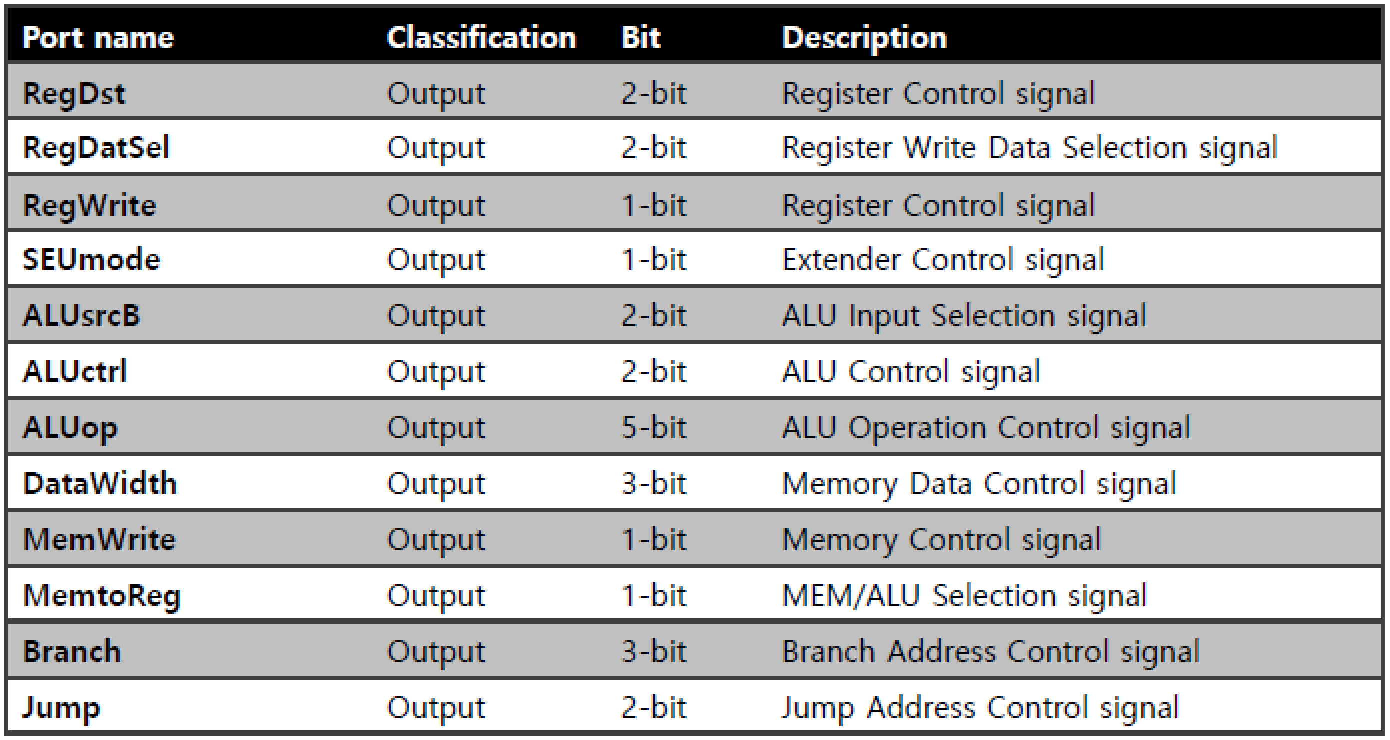


```
xxxxxx_xxxxxx_xxxxx  // 0x00 : sll
xxxxxx_xxxxxx_xxxxx  // 0x01 : srl
xxxxxx_xxxxxx_xxxxx  // 0x02 : sra
xxxxxx_xxxxxx_xxxxx  // 0x03 : sllv
xxxxxx_xxxxxx_xxxxx  // 0x04 : srlv
xxxxxx_xxxxxx_xxxxx  // 0x05 : srav
xxxxxx_xxxxxx_xxxxx  // 0x06 : jr
xxxxxx_xxxxxx_xxxxx  // 0x07 : jalr
xxxxxx_xxxxxx_xxxxx  // 0x08 : break
xxxxxx_xxxxxx_xxxxx  // 0x09 : mfhi
xxxxxx_xxxxxx_xxxxx  // 0x0a : mthi
xxxxxx_xxxxxx_xxxxx  // 0x0b : mflo
xxxxxx_xxxxxx_xxxxx  // 0x0c   mtlo
xxxxxx_xxxxxx_xxxxx  // 0x0d   mult
xxxxxx_xxxxxx_xxxxx  // 0x0e   multu
xxxxxx_xxxxxx_xxxxx  // 0x0f   div
```

Opcode_Function_Regimm

PLA_OR.txt

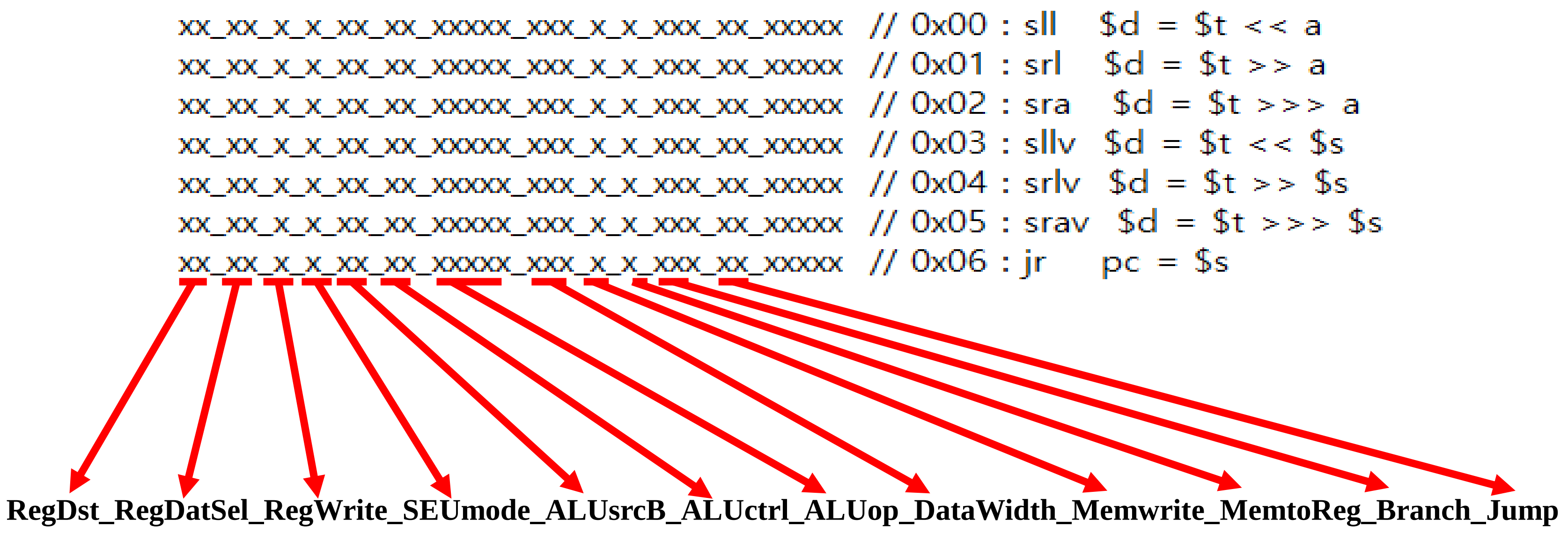
Table 3 – Consecutive Control Signal Configuration in PLA_OR.txt



Port name	Classification	Bit	Description
RegDst	Output	2-bit	Register Control signal
RegDatSel	Output	2-bit	Register Write Data Selection signal
RegWrite	Output	1-bit	Register Control signal
SEUmode	Output	1-bit	Extender Control signal
ALUsrcB	Output	2-bit	ALU Input Selection signal
ALUctrl	Output	2-bit	ALU Control signal
ALUop	Output	5-bit	ALU Operation Control signal
DataWidth	Output	3-bit	Memory Data Control signal
MemWrite	Output	1-bit	Memory Control signal
MemtoReg	Output	1-bit	MEM/ALU Selection signal
Branch	Output	3-bit	Branch Address Control signal
Jump	Output	2-bit	Jump Address Control signal

* **The last 5 bits are reserved. Set them as xxxxx.**

PLA_OR.txt

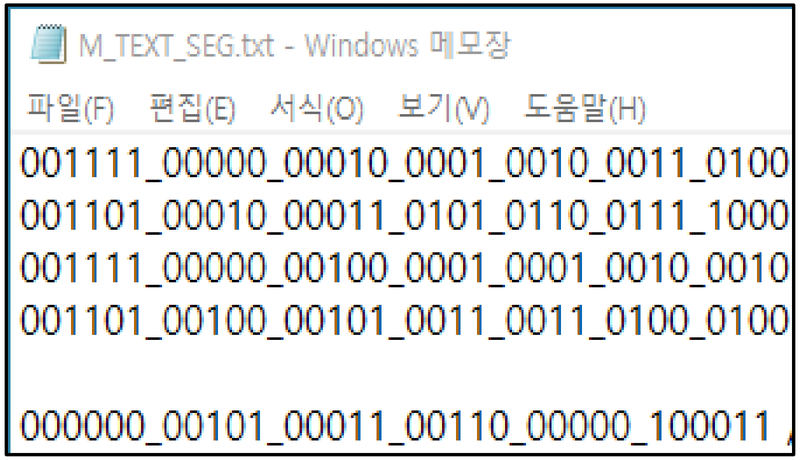


```
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x00 : sll   $d = $t << a
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x01 : srl   $d = $t >> a
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x02 : sra   $d = $t >>> a
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x03 : sllv  $d = $t << $s
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x04 : srlv  $d = $t >> $s
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x05 : srav  $d = $t >>> $s
xx_xx_x_x_xx_xx_xxxxx_xxx_x_x_xxx_xx_xxxxx  // 0x06 : jr    pc = $s
```

RegDst_RegDatSel_RegWrite_SEUmode_ALUsrcB_ALUctrl_ALUop_DataWidth_Memwrite_MemtoReg_Branch_Jump

Control Signals for SCPU.pdf

M_TEXT_SEG.txt


M_TEXT_SEG.txt - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

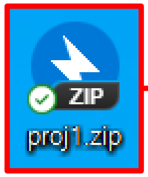
```
001111_00000_00010_0001_0010_0011_0100
001101_00010_00011_0101_0110_0111_1000
001111_00000_00100_0001_0001_0010_0010
001101_00100_00101_0011_0011_0100_0100

000000_00101_00011_00110_00000_100011
```

//lui $0 $2 0x1234 **lui : load upper immediate**
//ori $2 $3 0x5678 **ori : or immediate**
//lui $4 0x1122
//ori $5 $4 0x3344

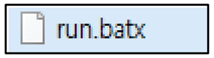
*You can change underbar(_) position for convenience

R-Type:	op	$rs	$rt	$rd	shamt	func
I-Type:	op	$rs	$rt	imm16		
J-Type:	op	imm26				

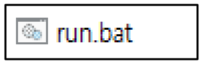
Simulation

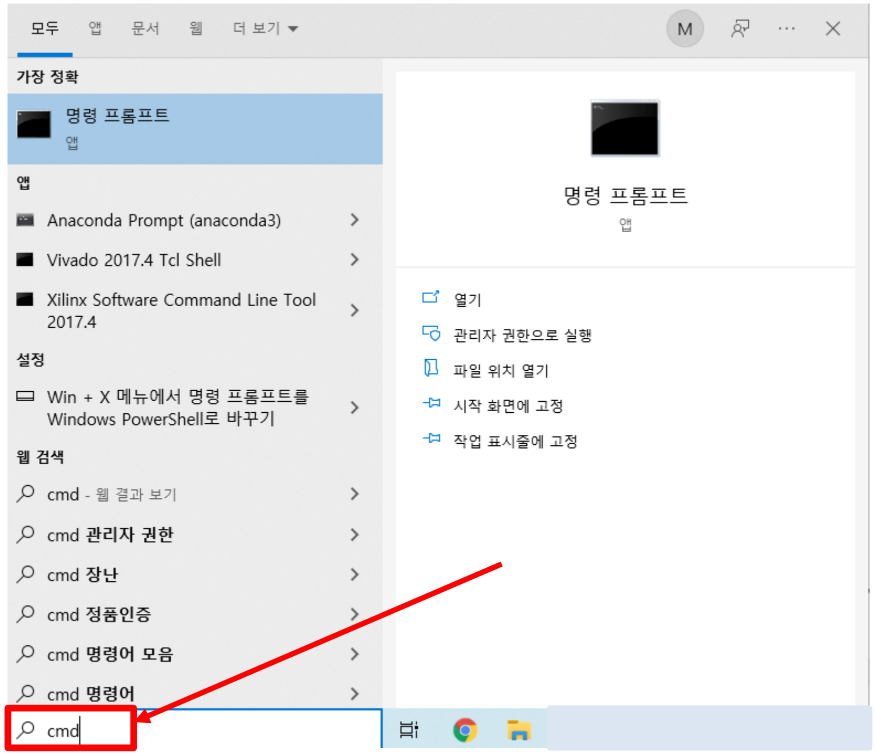


Unzip!



Rename!





가장 정확

명령 프롬프트
앱

앱

Anaconda Prompt (anaconda3)

Vivado 2017.4 Tcl Shell

Xilinx Software Command Line Tool 2017.4

설정

Win + X 메뉴에서 명령 프롬프트를 Windows PowerShell로 바꾸기

웹 검색

cmd - 웹 결과 보기

cmd 관리자 권한

cmd 장난

cmd 정품인증

cmd 명령어 모음

cmd 명령어

cmd

명령 프롬프트
앱

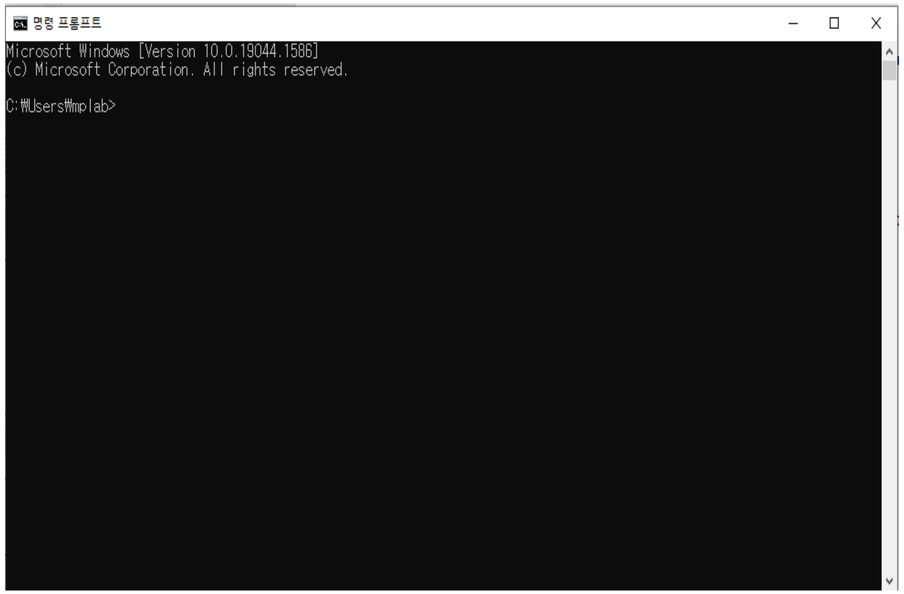
열기

관리자 권한으로 실행

파일 위치 열기

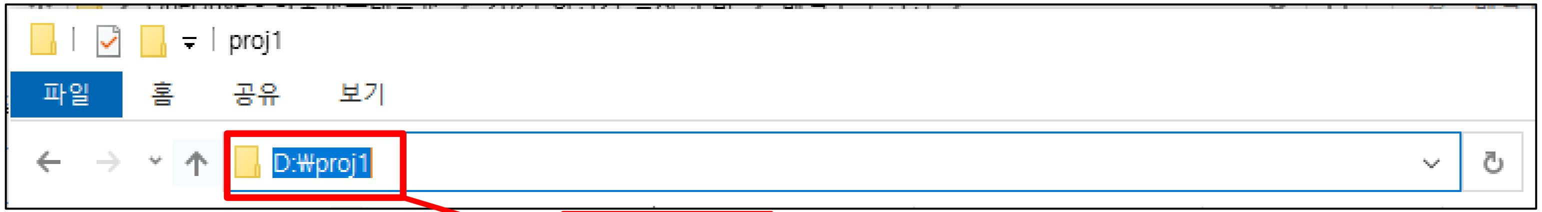
시작 화면에 고정

작업 표시줄에 고정



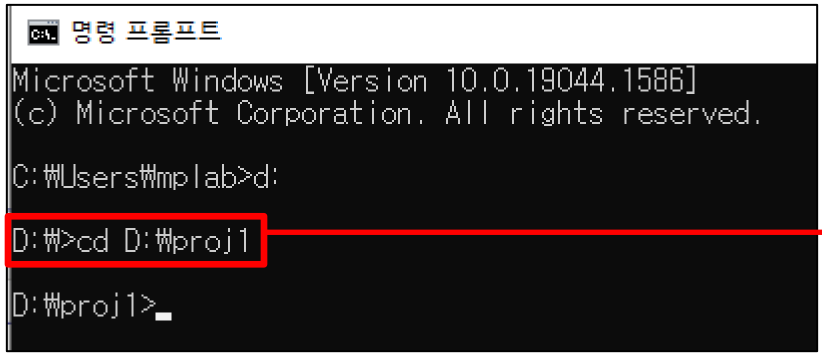
```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:₩Users₩mplab>
```

Simulation



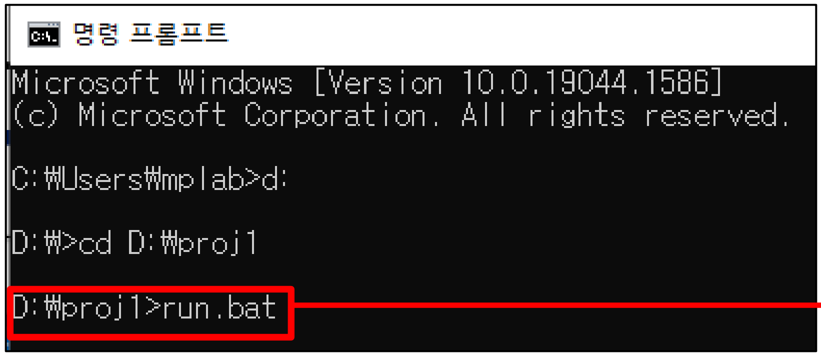
```
D:₩proj1
```

Copy!



```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:₩Users₩mplab>d:

D:₩>cd D:₩proj1

D:₩proj1>_
```

cd + paste root!

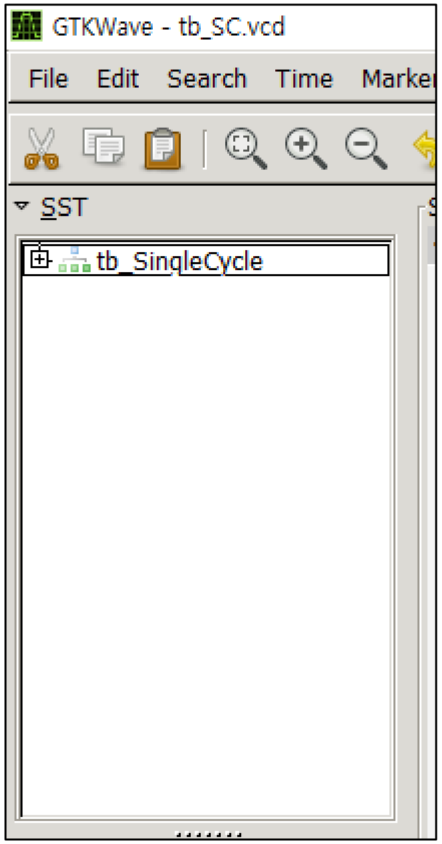


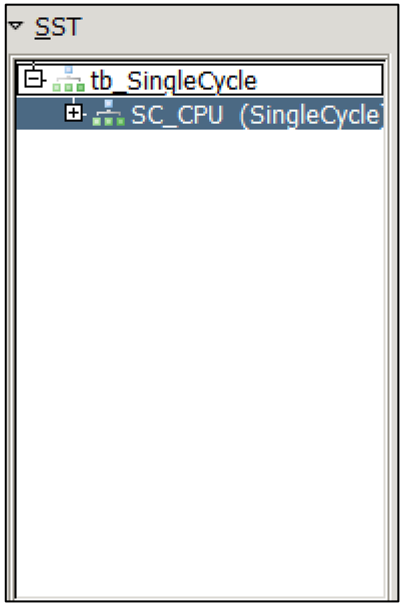
```
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:₩Users₩mplab>d:

D:₩>cd D:₩proj1

D:₩proj1>run.bat
```

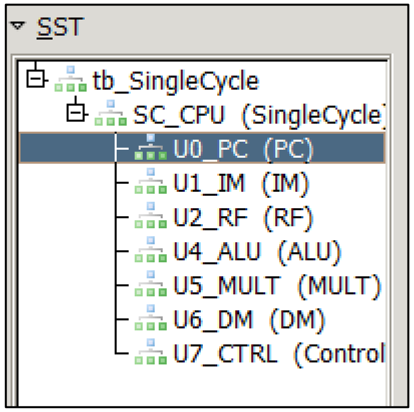
execute!

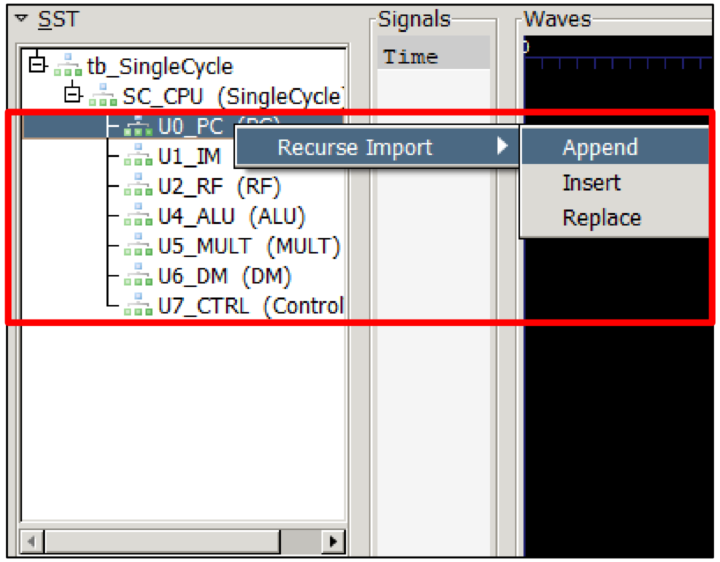
MPLAB
Media Processing Laboratory

Simulation

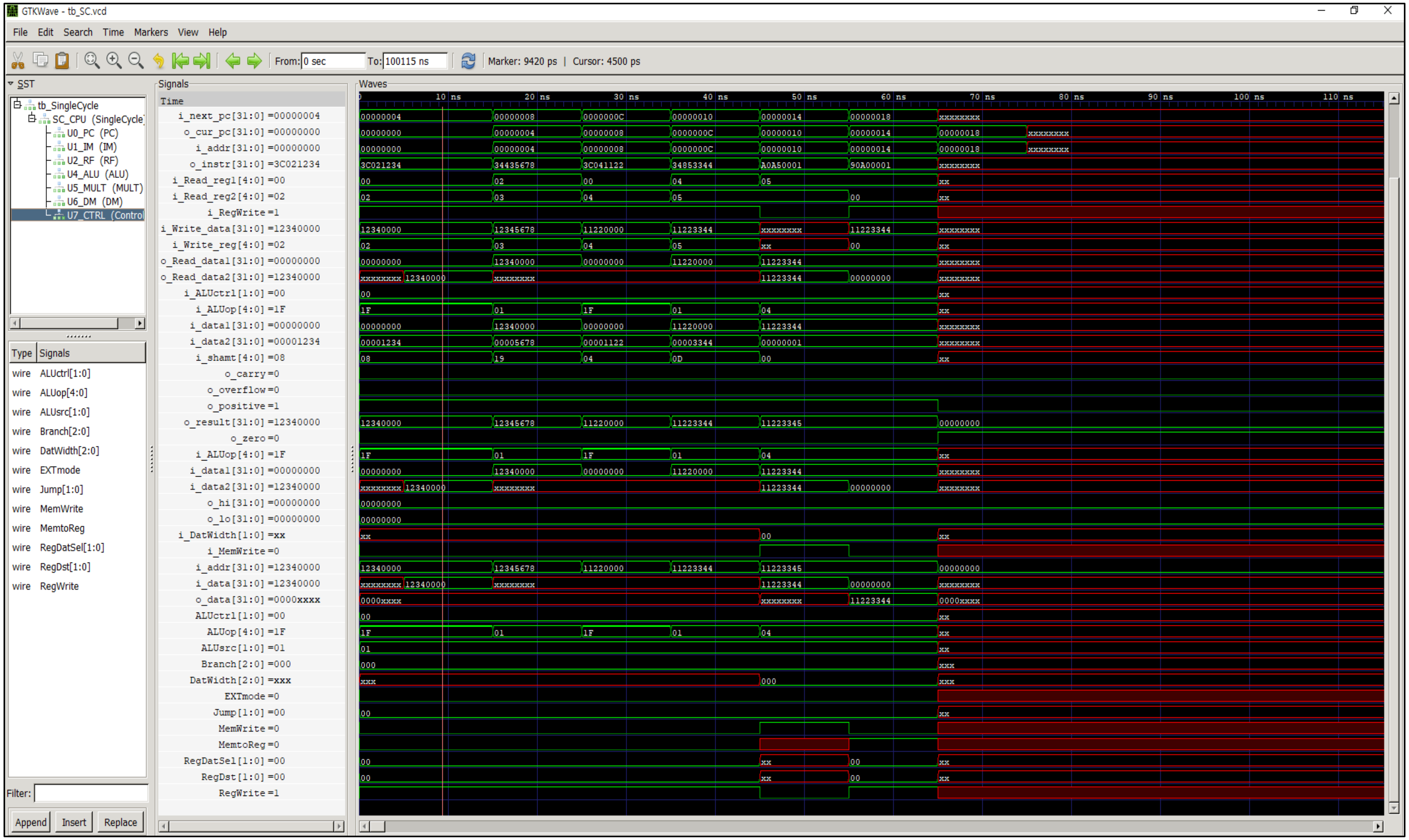

GTKWave - tb_SC.vcd
File Edit Search Time Marker
SST
tb_SingleCycle


SST
tb_SingleCycle
SC_CPU (SingleCycle)


SST
tb_SingleCycle
SC_CPU (SingleCycle)
U0_PC (PC)
U1_IM (IM)
U2_RF (RF)
U4_ALU (ALU)
U5_MULT (MULT)
U6_DM (DM)
U7_CTRL (Control)


SST
Signals
Waves
Time
tb_SingleCycle
SC_CPU (SingleCycle)
U0_PC (PC)
U1_IM
U2_RF (RF)
U4_ALU (ALU)
U5_MULT (MULT)
U6_DM (DM)
U7_CTRL (Control)
Recurse Import
Append
Insert
Replace

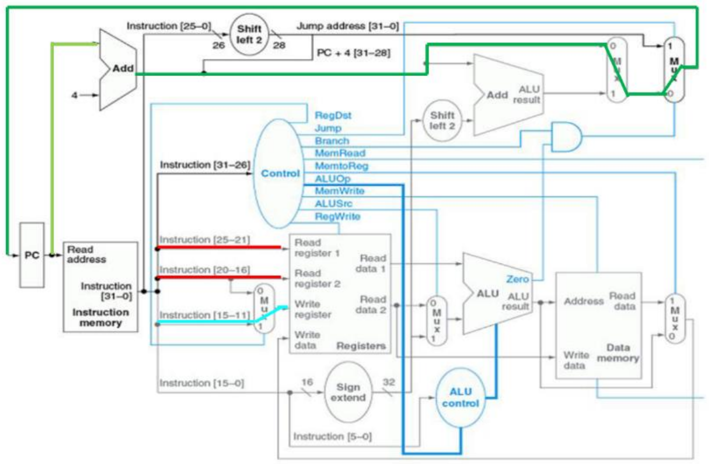
Simulation

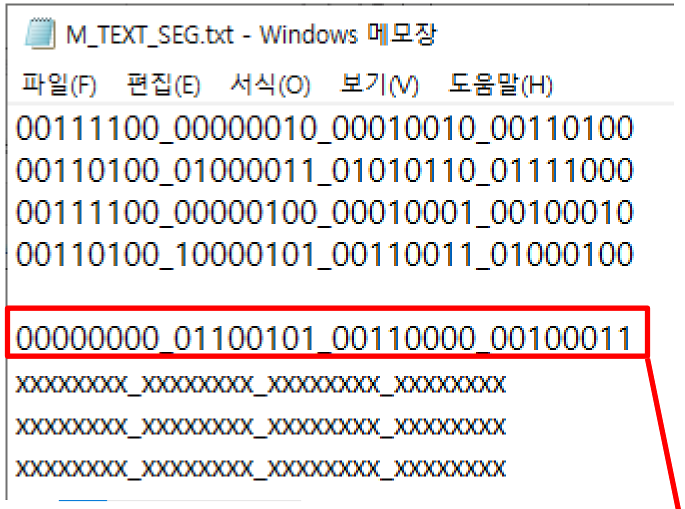


```
From: 0 sec   To: 100115 ns   Marker: 9420 ps  |  Cursor: 4500 ps

tb_SingleCycle
  SC_CPU (SingleCycle)
    U0_PC (PC)
    U1_IM (IM)
    U2_RF (RF)
    U4_ALU (ALU)
    U5_MULT (MULT)
    U6_DM (DM)
    U7_CTRL (Control)

Type  Signals
wire  ALUctrl[1:0]
wire  ALUop[4:0]
wire  ALUsrc[1:0]
wire  Branch[2:0]
wire  DatWidth[2:0]
wire  EXTmode
wire  Jump[1:0]
wire  MemWrite
wire  MemtoReg
wire  RegDatSel[1:0]
wire  RegDst[1:0]
wire  RegWrite

Time
i_next_pc[31:0] =00000004
o_cur_pc[31:0] =00000000
i_addr[31:0] =00000000
o_instr[31:0] =3C021234
i_Read_reg1[4:0] =00
i_Read_reg2[4:0] =02
i_RegWrite =1
i_Write_data[31:0] =12340000
i_Write_reg[4:0] =02
o_Read_data1[31:0] =00000000
o_Read_data2[31:0] =12340000
i_ALUctrl[1:0] =00
i_ALUop[4:0] =1F
i_data1[31:0] =00000000
i_data2[31:0] =00001234
i_shamt[4:0] =08
o_carry =0
o_overflow =0
o_positive =1
o_result[31:0] =12340000
o_zero =0
i_ALUop[4:0] =1F
i_data1[31:0] =00000000
i_data2[31:0] =12340000
o_hi[31:0] =00000000
o_lo[31:0] =00000000
i_DatWidth[1:0] =xx
i_MemWrite =0
i_addr[31:0] =12340000
i_data[31:0] =12340000
o_data[31:0] =0000xxxx
ALUctrl[1:0] =00
ALUop[4:0] =1F
ALUsrc[1:0] =01
Branch[2:0] =000
DatWidth[2:0] =xxx
EXTmode =0
Jump[1:0] =00
MemWrite =0
MemtoReg =0
RegDatSel[1:0] =00
RegDst[1:0] =00
RegWrite =1
```

MPLAB
Media Processing Laboratory

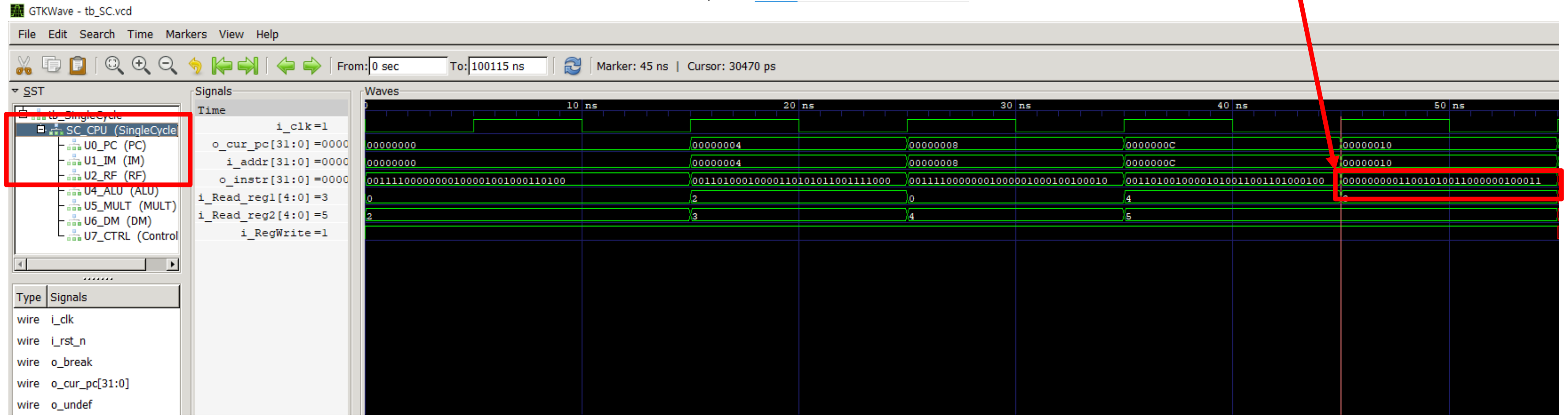
Simulation




M_TEXT_SEG.txt - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
00111100_00000010_00010010_00110100
00110100_01000011_01010110_01111000
00111100_00000100_00010001_00100010
00110100_10000101_00110011_01000100

00000000_01100101_00110000_00100011
XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
XXXXXXXX_XXXXXXXX_XXXXXXXX_XXXXXXXX
```


GTKWave - tb_SC.vcd

File Edit Search Time Markers View Help

From: 0 sec To: 100115 ns Marker: 45 ns | Cursor: 30470 ps

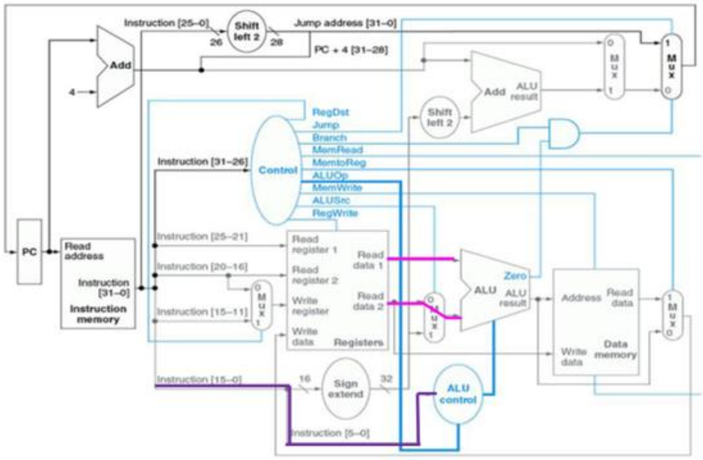
SST
- tb_SingleCycle
  - SC_CPU (SingleCycle)
    - U0_PC (PC)
    - U1_IM (IM)
    - U2_RF (RF)
    - U4_ALU (ALU)
    - U5_MULT (MULT)
    - U6_DM (DM)
    - U7_CTRL (Control)

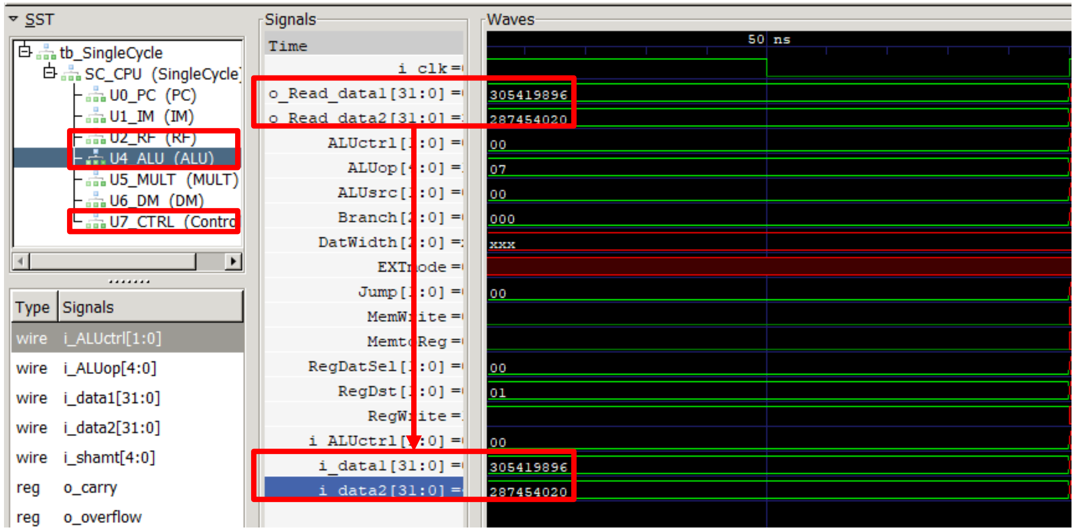
Type	Signals
wire	i_clk
wire	i_rst_n
wire	o_break
wire	o_cur_pc[31:0]
wire	o_undef

```
Signals                    Waves
Time                       0          10 ns          20 ns          30 ns          40 ns          50 ns
i_clk=1
o_cur_pc[31:0]=0000        00000000          00000004          00000008          0000000C          00000010
i_addr[31:0]=0000          00000000          00000004          00000008          0000000C          00000010
o_instr[31:0]=0000         00111100000000100001001000110100  00110100010000110101011001111000  00111100000001000001000100100010  00110100100001010011001101000100  00000000011001010011000000100011
i_Read_reg1[4:0]=3         0                 2                 0                 4
i_Read_reg2[4:0]=5         2                 3                 4                 5
i_RegWrite=1
```

MPLAB
Media Processing Laboratory

Simulation



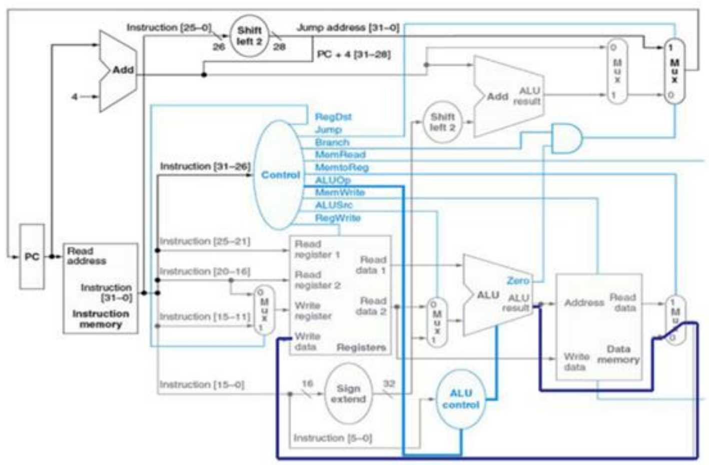


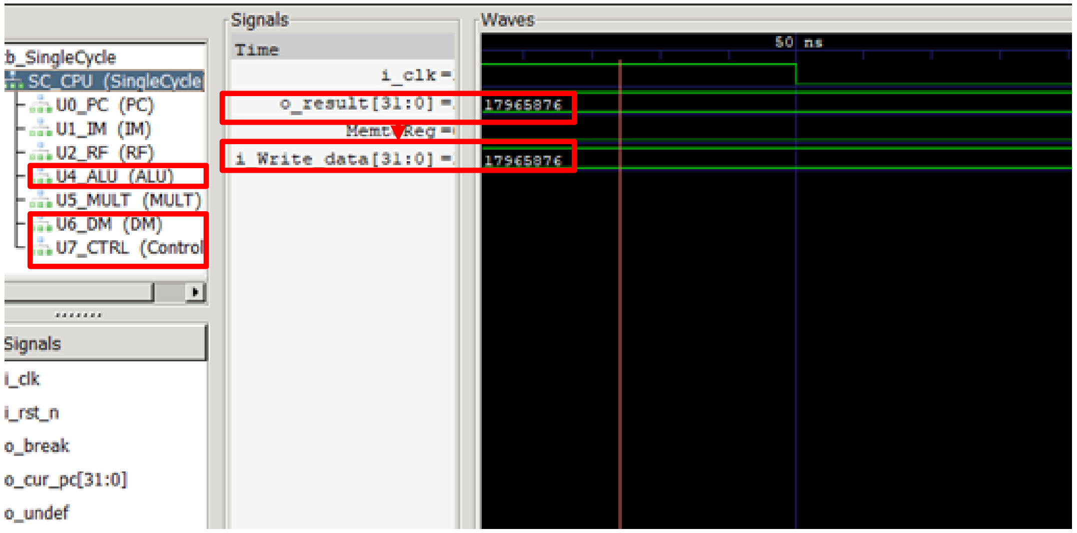
```
SST
tb_SingleCycle
  SC_CPU (SingleCycle)
    U0_PC (PC)
    U1_IM (IM)
    U2_RF (RF)
    U4_ALU (ALU)
    U5_MULT (MULT)
    U6_DM (DM)
    U7_CTRL (Contro

Type  Signals
wire  i_ALUctrl[1:0]
wire  i_ALUop[4:0]
wire  i_data1[31:0]
wire  i_data2[31:0]
wire  i_shamt[4:0]
reg   o_carry
reg   o_overflow
```

```
Signals                      Waves
Time                                          50 ns
           i_clk =
o_Read_data1[31:0] =         305419896
o_Read_data2[31:0] =         287454020
     ALUctrl[1:0] =          00
       ALUop[4:0] =          07
      ALUsrc[1:0] =          00
      Branch[2:0] =          000
   DatWidth[2:0] =           xxx
         EXTmode =
        Jump[1:0] =          00
        MemWrite =
        MemtoReg =
   RegDatSel[1:0] =          00
      RegDst[1:0] =          01
        RegWrite =
   i_ALUctrl[1:0] =          00
    i_data1[31:0] =          305419896
    i_data2[31:0] =          287454020
```

MPLAB Media Processing Laboratory

Simulation





```
b_SingleCycle
  SC_CPU (SingleCycle
    U0_PC (PC)
    U1_IM (IM)
    U2_RF (RF)
    U4_ALU (ALU)
    U5_MULT (MULT)
    U6_DM (DM)
    U7_CTRL (Control

Signals
i_clk
i_rst_n
o_break
o_cur_pc[31:0]
o_undef
```

```
Signals                      Waves
Time                                       50 ns
              i_clk =
     o_result[31:0] =        17965876
           MemtoReg =
 i_Write_data[31:0] =        17965876
```

END