CSL
Computer Systems Laboratory

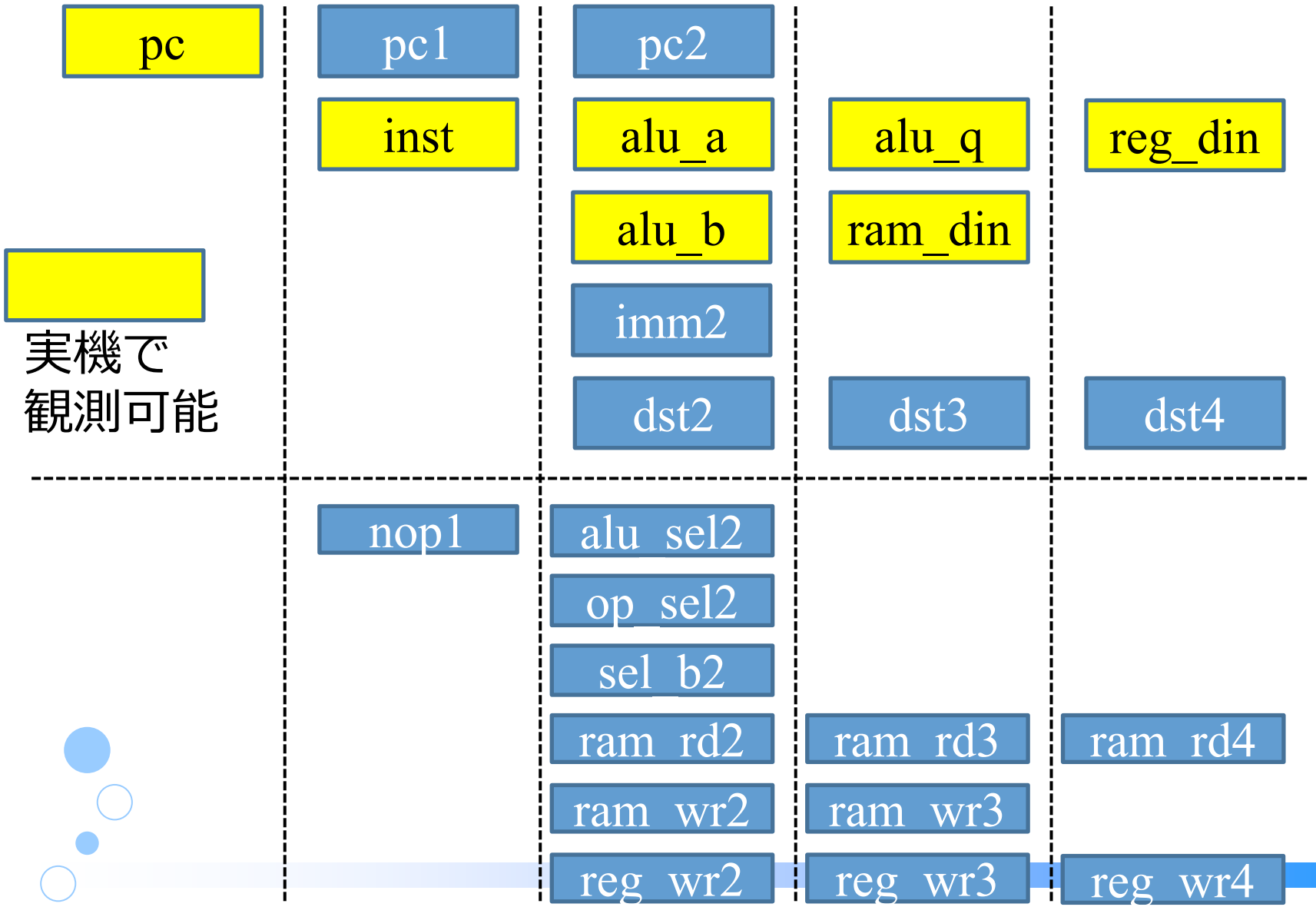
情報科学演習
プロセッサ演習
～第３週～

情報科学科
コンピュータシステム研究室

シミュレータと実機の主な違い

	Mars	実機
命令メモリ（text）	0040 0000 ～ 0FFF FFFC	0040 0000 ～ 0040 07FC （512語）
データメモリ（extern）	1000 0000 ～ 1000 FFFC	なし
データメモリ（data）	1001 0000 ～ 1003 FFFC	1001 0000 ～ 1001 7FFC （8192語）
データメモリ（heap）	1004 0000 ～	なし
$sp の初期値	7FFF EFFC	不定値（初期化必要）
$gp の初期値	1000 8000	不定値（初期化必要）
他のレジスタの初期値	0000 0000	不定値（初期化必要）
syscall	$v0 = 10 で終了	$v0 任意の値で CPU一時停止
データハザード	該当なし	スイッチで機構の有無を選択
フォーワーディング	該当なし	スイッチで機構の有無を選択
遅延分岐	オプションで選択	スイッチで機構の有無を選択

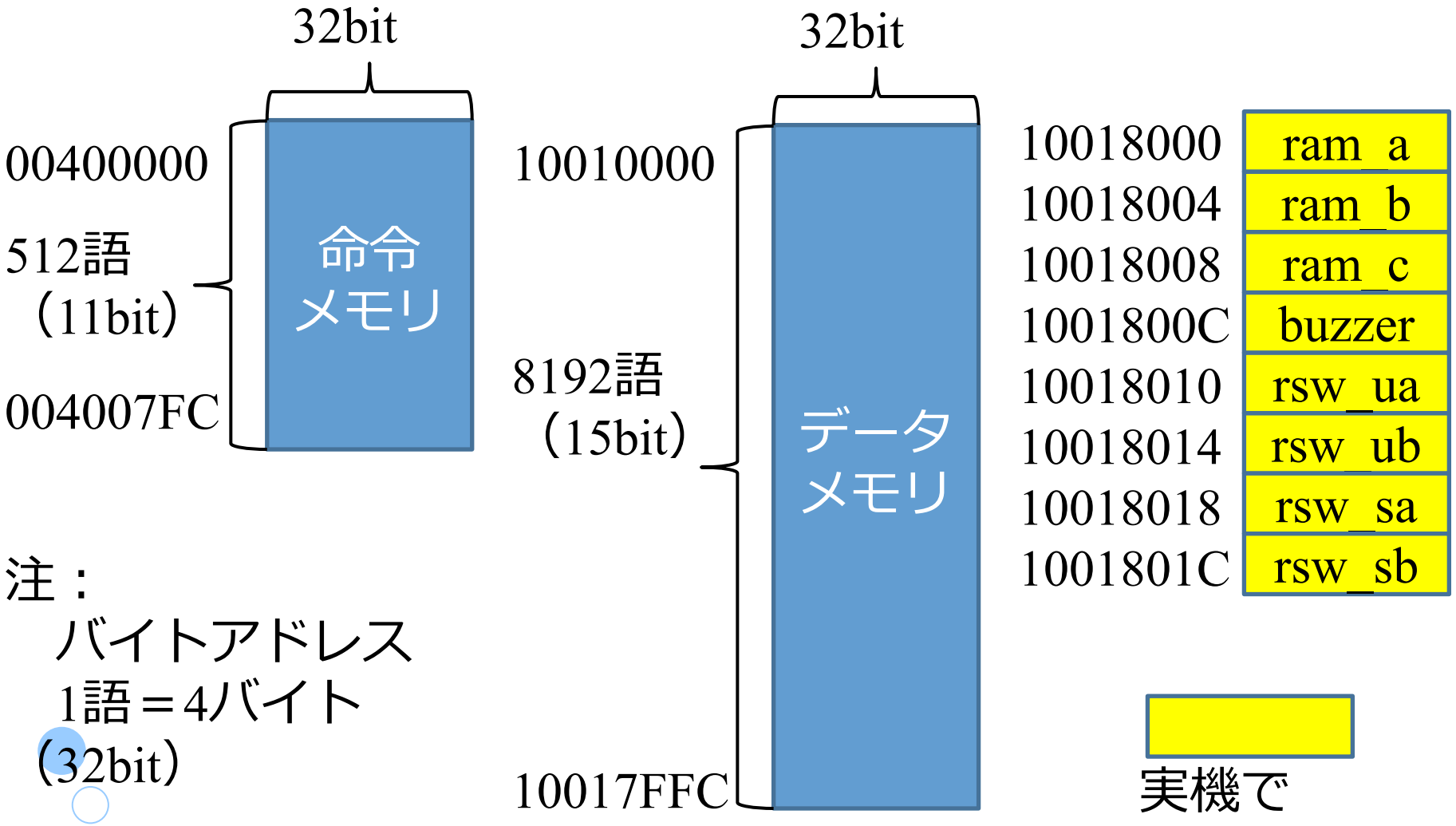
プロセッサの内部信号


pc
pc1
pc2
inst
alu_a
alu_q
reg_din
alu_b
ram_din
imm2
実機で
観測可能
dst2
dst3
dst4
nop1
alu_sel2
op_sel2
sel_b2
ram_rd2
ram_rd3
ram_rd4
ram_wr2
ram_wr3
reg_wr2
reg_wr3
reg_wr4

観測可能な内部信号

pc	プログラムカウンタ
inst	機械語命令
alu_a	ALU第1入力データ（フォワーディング後）
alu_b	ALU第2入力データ（フォワーディング後）
alu_q	ALU演算結果 （ロード・ストアではメモリ・アドレス）
ram_din	メモリ書き込みデータ（ストアで使用）
reg_din	レジスタ書き込みデータ
cnt	クロックカウンタ（top.vで定義）

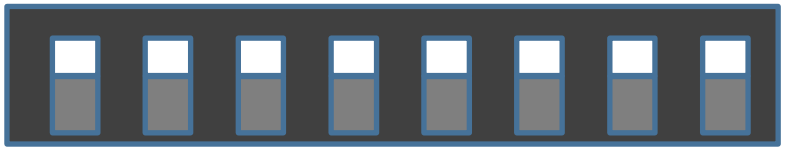
実機のメモリ・マップ



注：
バイトアドレス
1語＝4バイト
(32bit)

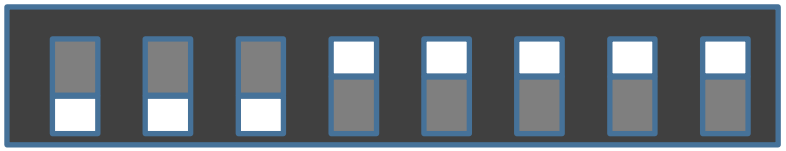
ディップスイッチと ロータリースイッチの仕様

DIP_A



7 6 5 4 3 2 1 0

DIP_B



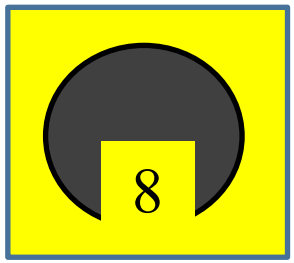
7 6 5 4 3 2 1 0

上側：OFF、下側：ON

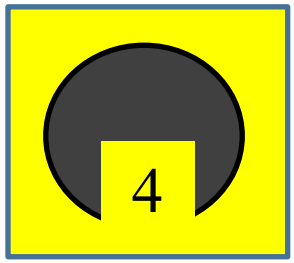
DIP_B-7：ハザード検出機構

DIP_B-6：フォワーディング機構

DIP_B-5：遅延分岐機構



HEX_A



HEX_B

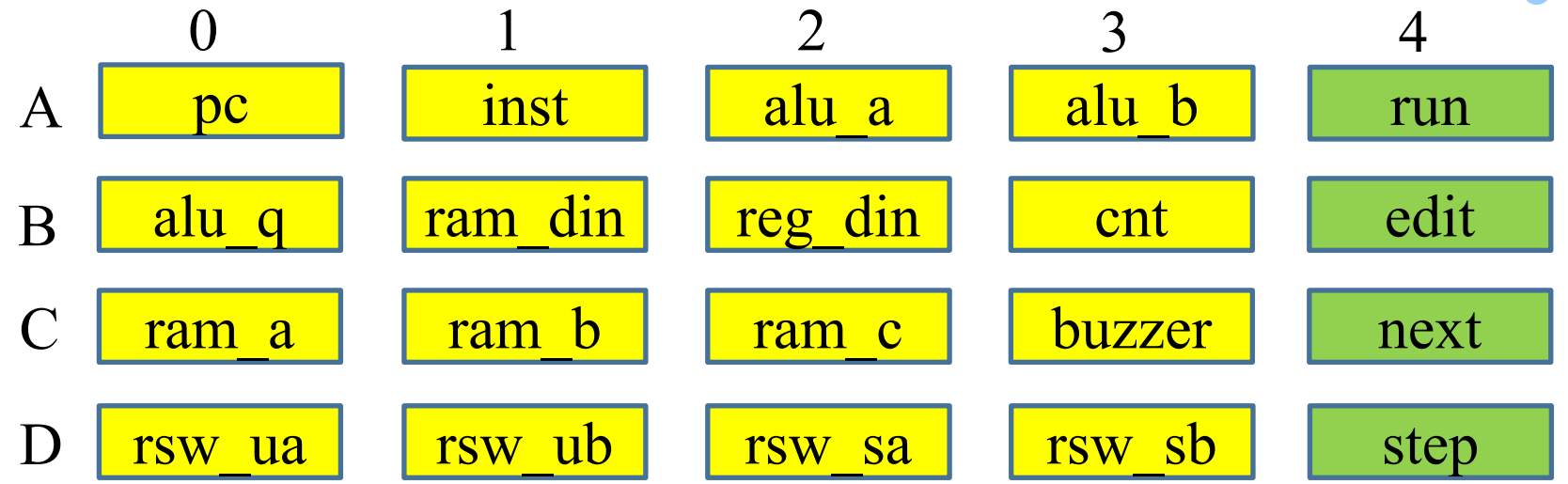
設定値と対応する値を入力

符号なし／符号ありの
32ビットの値に拡張

観測可能な入出力ポート

ram_a	メモリA	lwで読み出し可、swで書き込み可 キーボード操作で書き込み可
ram_b	メモリB	
ram_c	メモリC	
buzzer	ブザー	
rsw_ua	ロータリースイッチA（符号なし）	lwで 読み出し可
rsw_ub	ロータリースイッチB（符号なし）	
rsw_sa	ロータリースイッチA（符号あり）	
rsw_sb	ロータリースイッチB（符号あり）	

キーボードの配置（観測モード）



	格納されている値を表示
run	実行・停止モードの切り替え （初期状態は停止モード）
edit	観測モードから編集モードへ移行
next	cnt, pc, inst, … , reg_din, cnt の順に表示を変更
step	1パルスのクロックを出力してステップ実行 （実行モード時は停止モードへ強制移行）

キーボードの配置（編集モード）

	0	1	2	3	4
A	0	1	2	3	write
B	4	5	6	7	view
C	8	9	A	B	next
D	C	D	E	F	bs

	値を1桁入力
write	入力した値をnextで指定した場所に書き込み
view	編集モードから観測モードへ移行
next	rama, ramb, ramc, buzzer の順に書き込み先を変更
bs	値を1桁削除（バックスペース）

LEDの表示内容

0　　点灯：編集モード、消灯：観測モード
1　　点灯：実行モード、消灯：停止モード
2　　クロック信号
3　　リセット信号
4～7
　観測モードのとき
　　0000:pc,　　0001:inst,　0010:alu_a,　0011:alu_b
　　0100:alu_q,　0101:ram_din,0110:reg_din,0111:cnt
　　1000:ram_a,　1001:ram_b,　1010:ram_c,　1011:buzzer
　　1100:rsw_ua, 1101:rsw_ub, 1110:rsw_sa, 1111:rsw_sb
　編集モードのとき
　　0000:ram_a,　0001:ram_b,　0010:ram_c,　0011:buzzer

クロック周波数の設定

ロータリースイッチCK_DIVの設定値と周波数の関係

設定値	周波数	設定値	周波数
0	40MHz	8	9.8kHz
1	20MHz	9	4.9kHz
2	10MHz	A	2.44kHz
3	5MHz	B	1.22kHz
4	1.25MHz	C	610Hz
5	312.5kHz	D	305Hz
6	78.1kHz	E	1.0Hz
7	19.5kHz	F	未使用

クロック周波数の設定

```
...
    addiu       $a0, $zero, 95
    sw          $a0, 396($s0)
    li          $v0, 2
    syscall                 # 一時停止
    nop
Sort0:
########################################
...
########################################
Done:
    nop
    li          $s3,  0x10018000
    addiu       $t5,  $zero,200
    sw          $t5,  12($s3)
    li          $v0,  1
    syscall                 # 一時停止
```

実機はここで一時停止
runキーを押すと再開

以降はTAが操作して実行結果を確認

ROMファイルの作成

■アセンブリプログラムの出力

- File > Dump Memory
- Dump Format > Text/Data Segment Window
- 「rom.txt」(ファイル名)で保存

■Verilogソースファイルへの変換

- perl　rom.pl　rom.txt

※ bubble100.asm と Verilogソースファイル群は day3のファイルを使用すること。
（実機の動作のために微修正をしている）

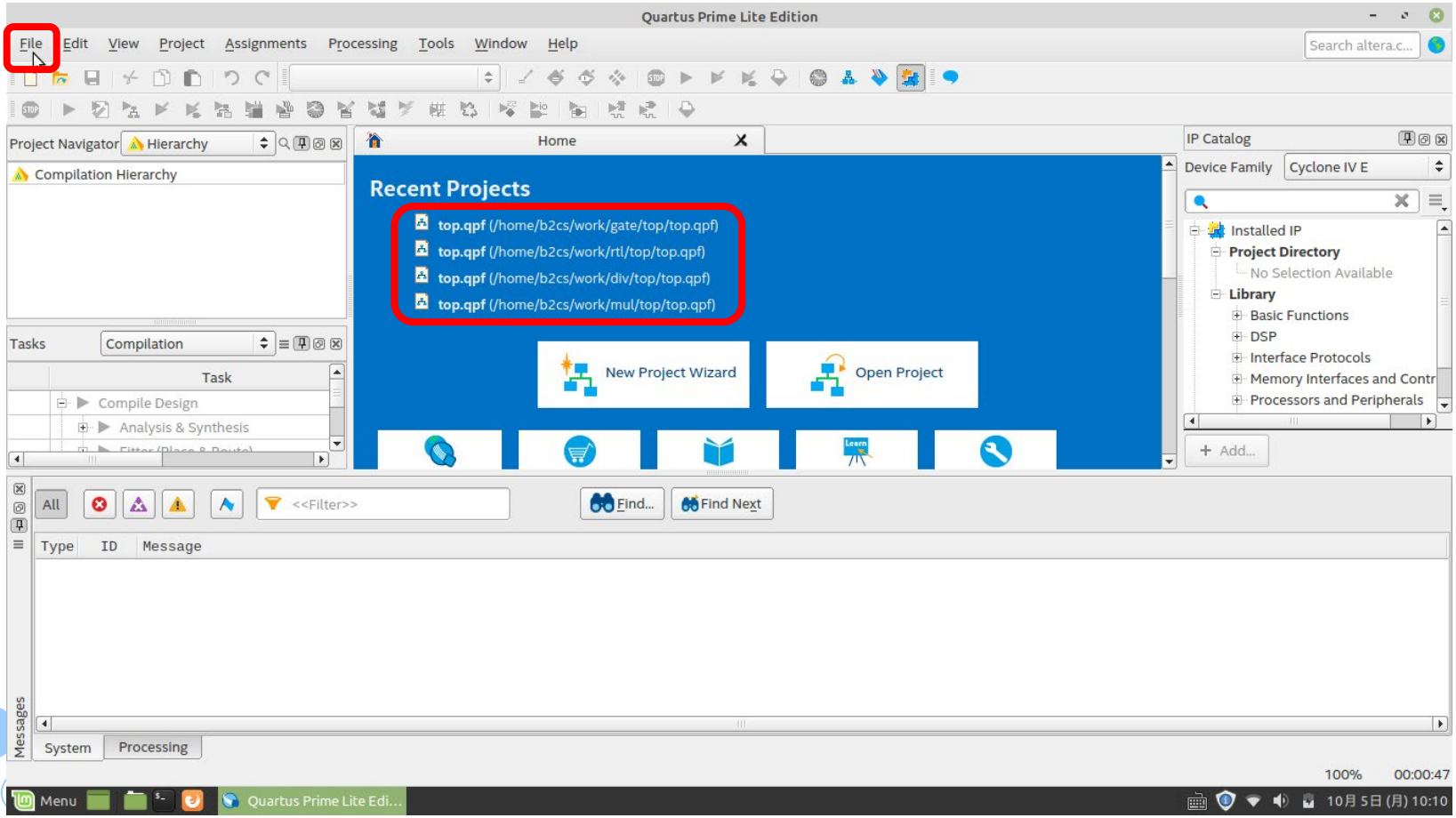
◎ Quartusの起動

デスクトップ上のQuartusを起動

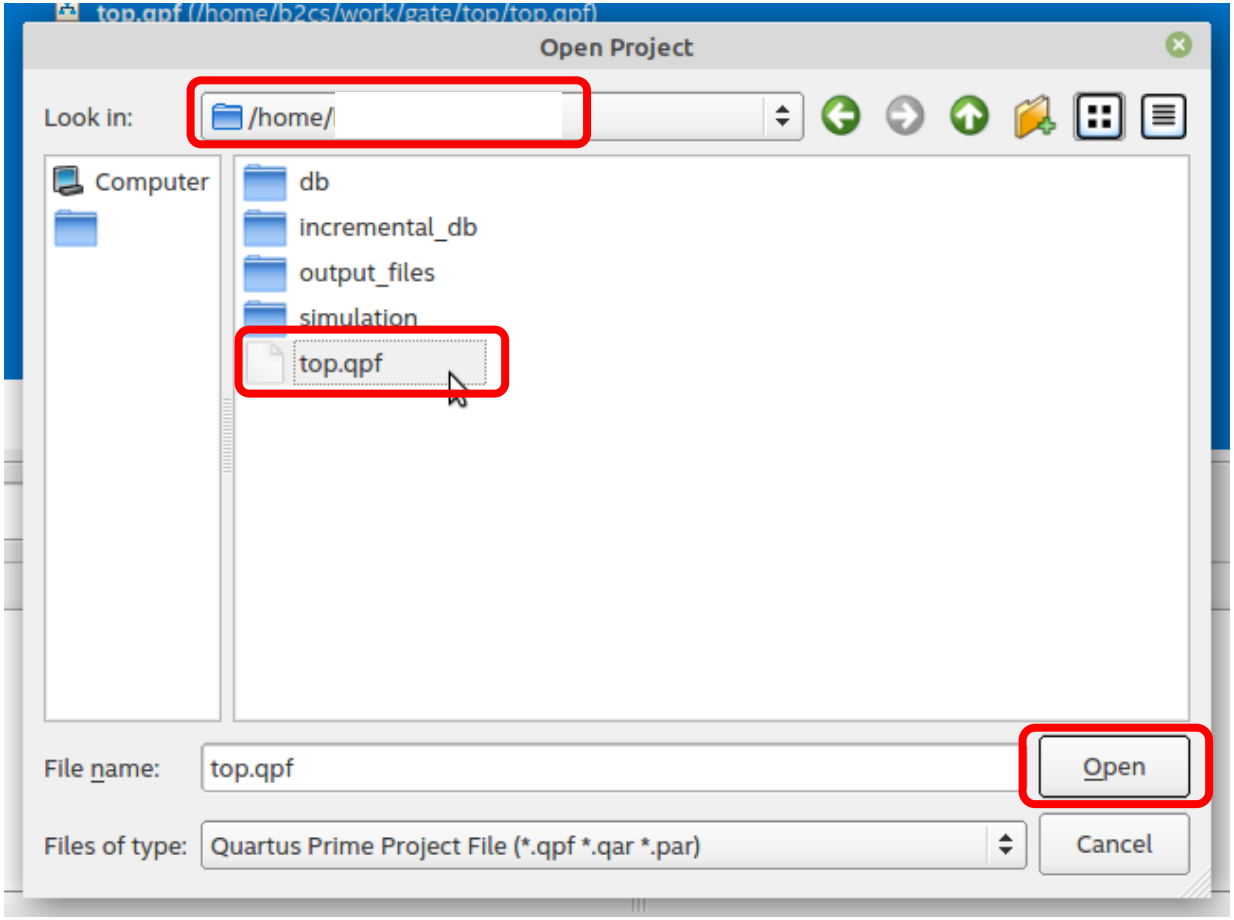
◎ プロジェクトの開始

File ➔ Open Project (Ctrl+J)を選択

Recent Projectsにある場合は、そこから選択しても可

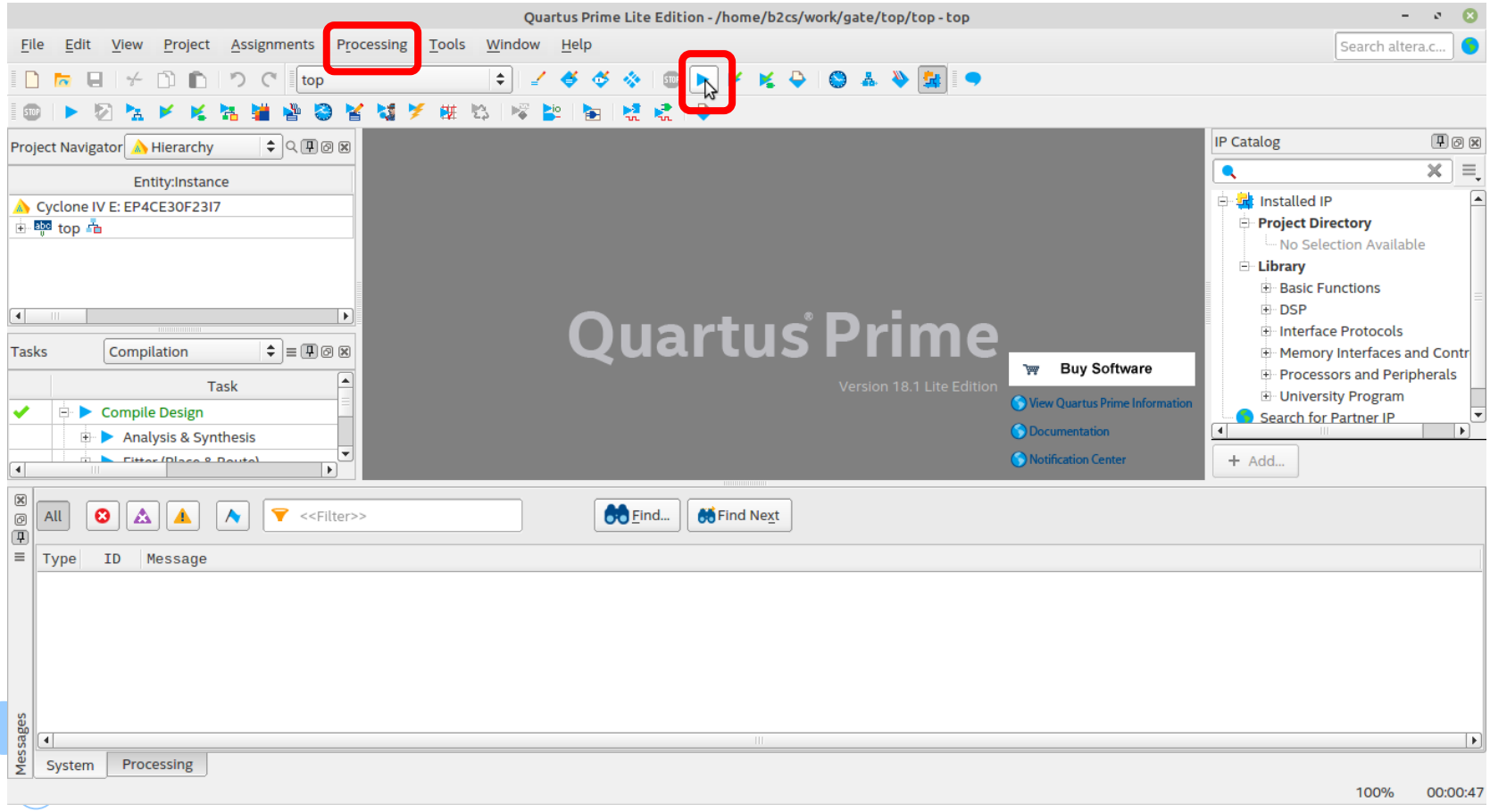


/home/b3cs/work/mips/top.qpfを選択



◎ ソースファイルのコンパイル

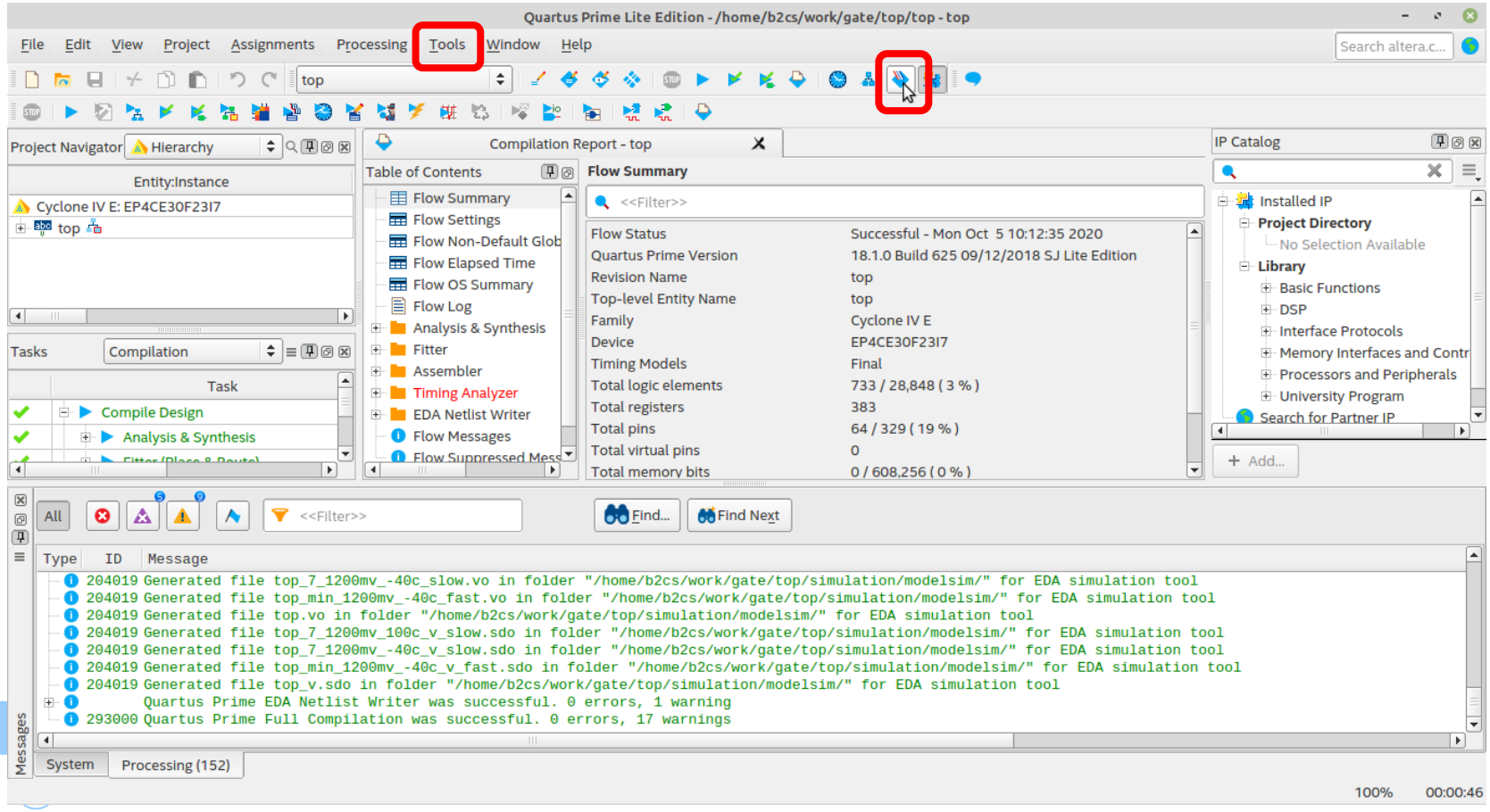
Processing ➔ Start Compilation (Ctrl+L)を選択
以下のアイコンを選択しても可



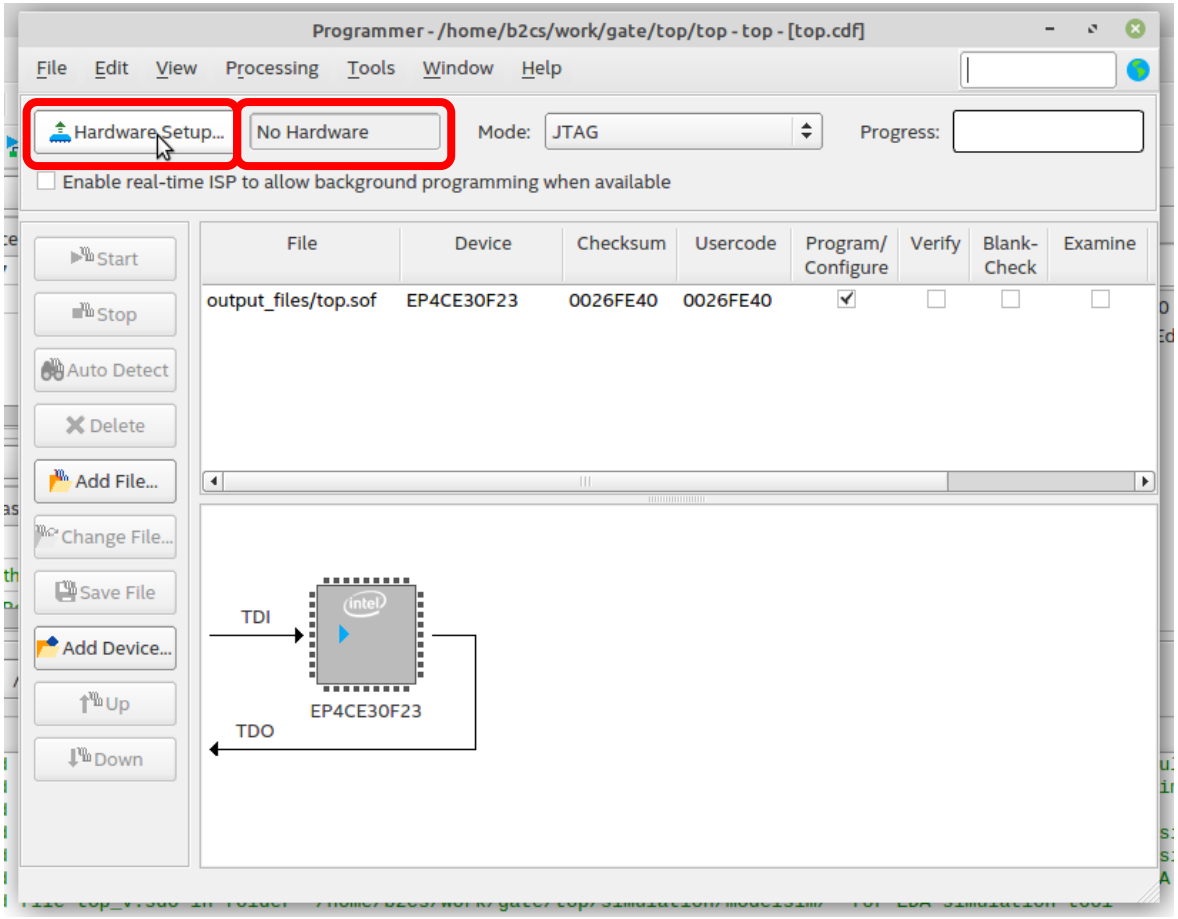
◎ 実機への実装

Tool ➔ Programmerを選択

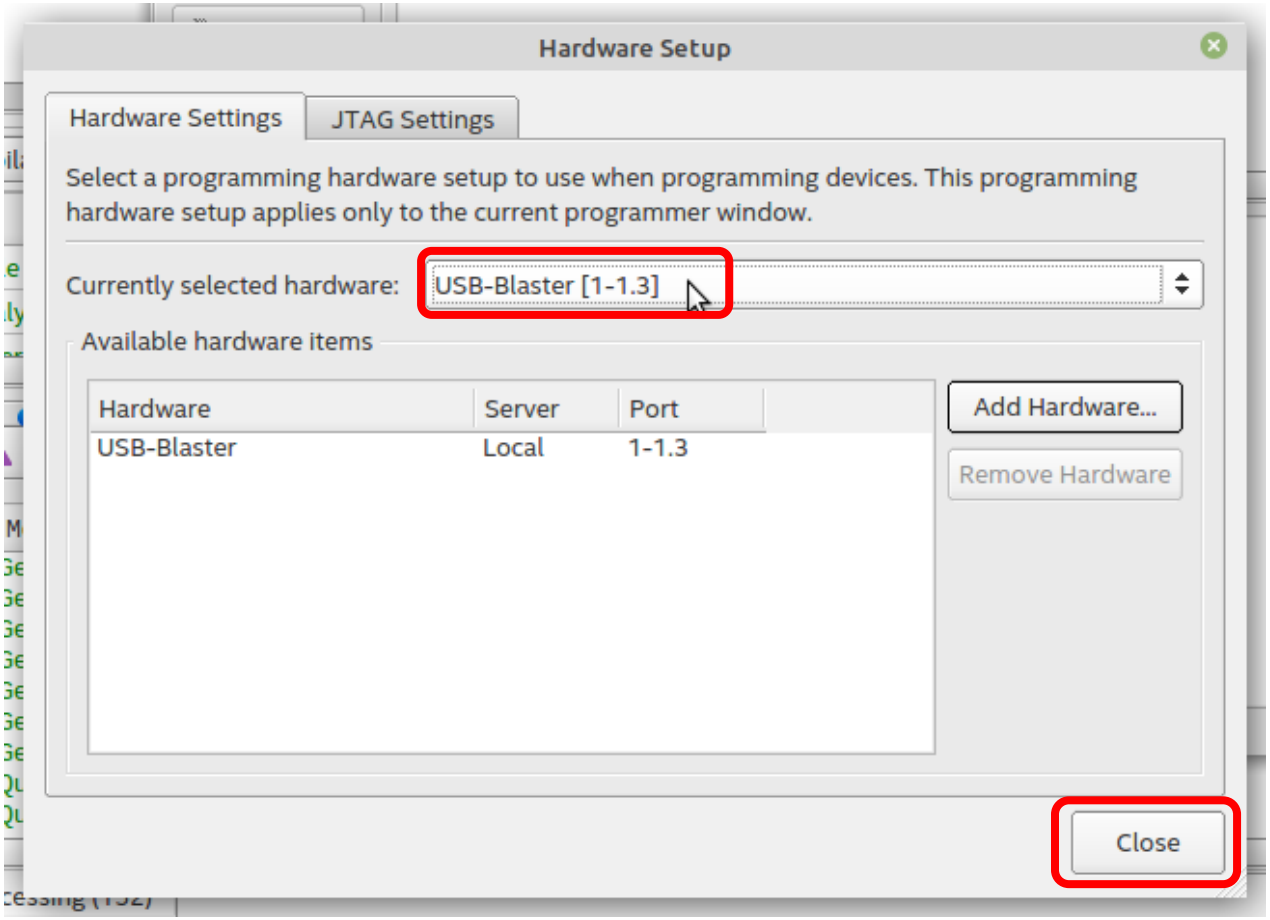
以下のアイコンを選択しても可



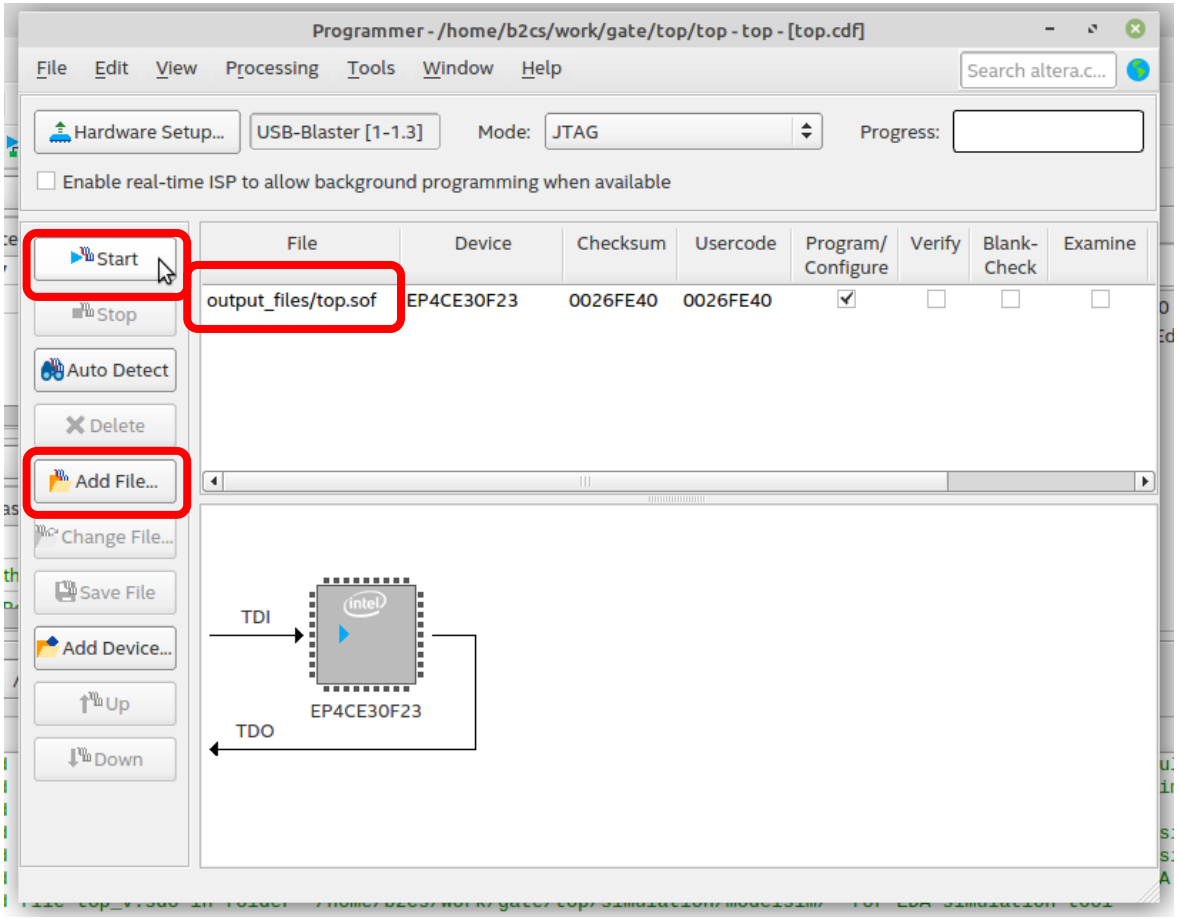
No Hardwareとなっている場合は
USB-BlasterをUSBポートに接続した状態で
Hardware Setupを選択



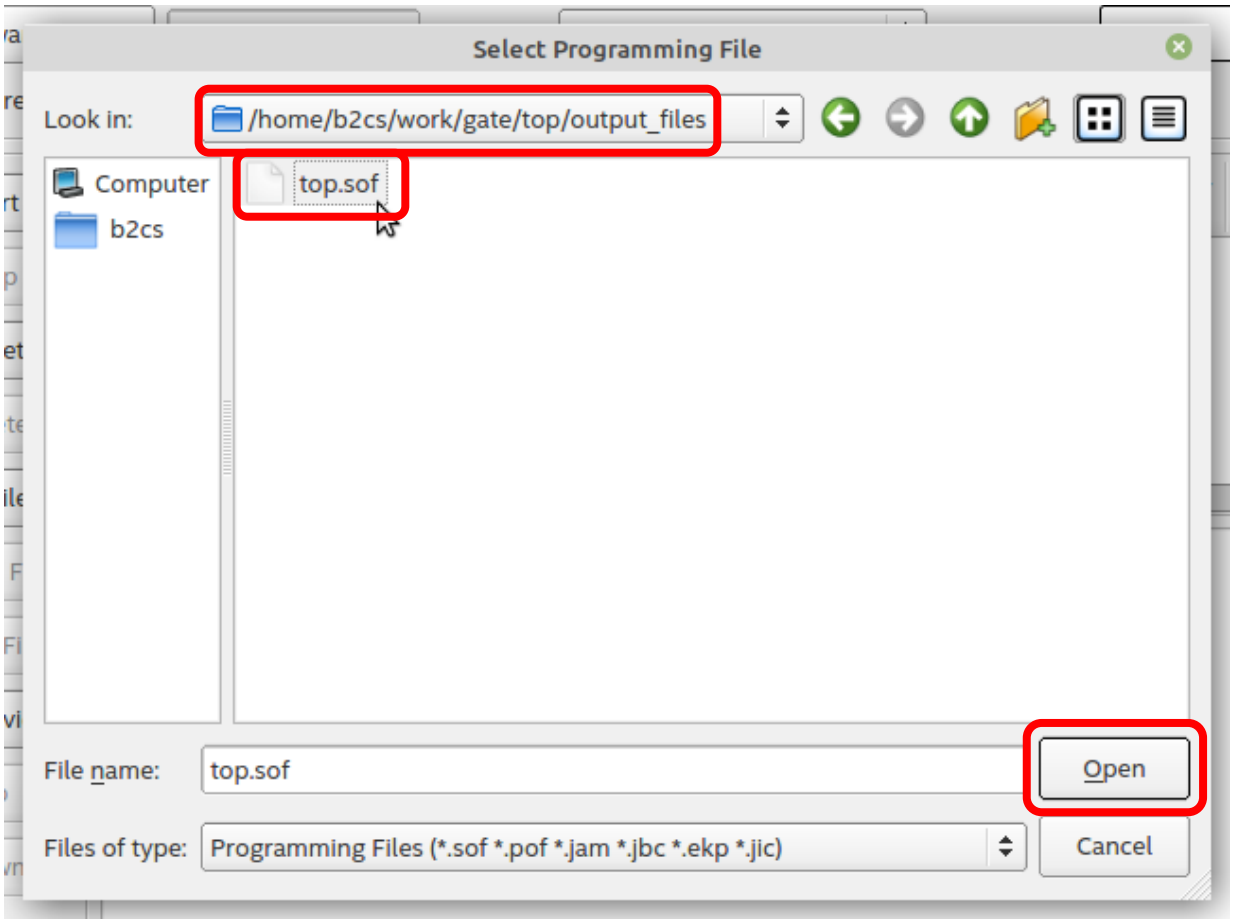
USB-Blasterを選択してCloseを選択
（前の画面で選択済みならこの操作は不要）



output_files/top.sofが選択済みならStartを選択
未選択ならAdd Fileを選択してからStartを選択



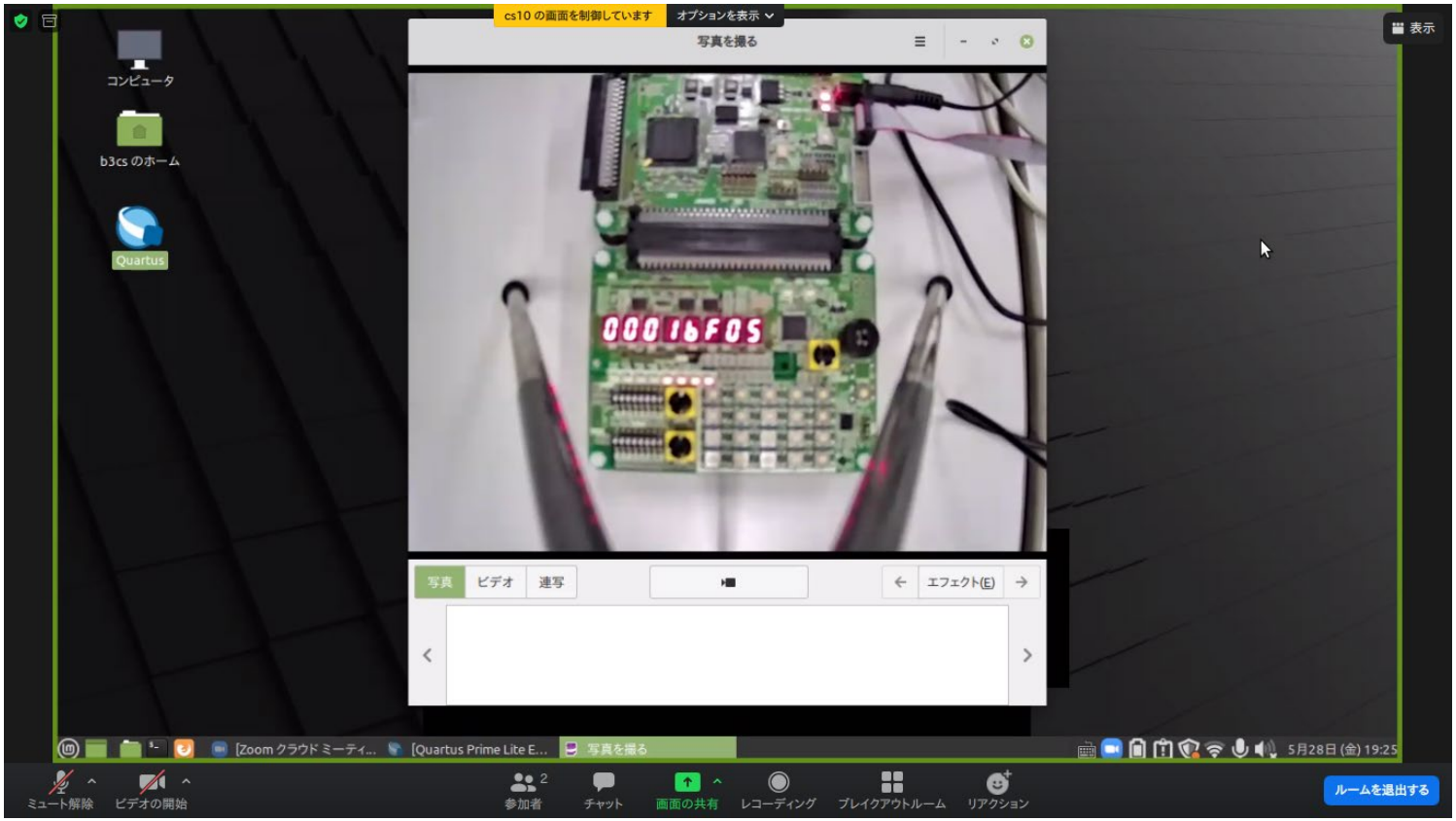
output_files/top.sofを選択してOpenを選択
（前の画面で選択済みならこの操作は不要）



実行結果の確認

7SEGLED上に表示されるクロック・サイクル数を記録

※ 参考記録： $1BF06_{(16)}$ = 114,438クロック

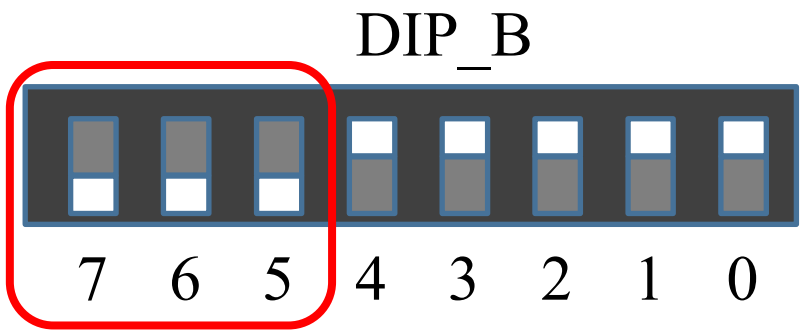


第3週の課題

ソースファイル	ハザード検出	フォワーディング	遅延分岐
pipeline2.asm	off	off	off
hazard2.asm	on	on/off	off
branch2.asm	on	on	on/off

- 上記の各設定に関して、第2週のHDLシミュレーションの結果と対応しているか？異なる動作をする場合はあるか？
- HDLソースファイル群は day3のものを使用すること。

7: ハザード検出機構
6: フォワーディング機構
5: 遅延分岐機構



第３週の課題

- bubble100.asm: ソートのプログラムを実機上で実行
  - プログラムはday3のものを使用すること。
  - クロック・サイクル数の結果を確認
  - ソートの結果を確認（TAがチェック）
  - 遅延分岐をオンにして、高速なプログラムを検討

- ???sort.asm: より高速なソートアルゴリズムの実装

プログラム作成に関する注意

- レジスタ・ファイルの初期値は全て不定値
  - ➔ 初期化の命令の追加が必要な場合あり
- ロードの結果を次命令で使用するとハザード
  （直前の命令の結果を分岐命令が使う場合も）
  - ➔ 実行結果に影響が無い命令（nop含む）を配置
- 分岐命令の直後はハザード or 遅延スロット
  - ➔ 実行結果に影響が無い命令（nop含む）を配置

バブルソート（隣接データを比較）

1	5	2	4	3	6
1	5	2	4	6	3
1	5	2	6	4	3
1	5	6	2	4	3
1	6	5	2	4	3
6	1	5	2	4	3
6	1	5	2	4	3
6	1	5	4	2	3
6	1	5	4	2	3
6	5	1	4	2	3
6	5	1	4	3	2
6	5	1	4	3	2
6	5	4	1	3	2
6	5	4	1	3	2
6	5	4	3	1	2
6	5	4	3	2	1

選択ソート（最大値から先頭へ）

1	3	6	5	4	2
6	3	1	5	4	2
6	5	1	3	4	2
6	5	4	3	1	2
6	5	4	3	1	2
6	5	4	3	2	1

挿入ソート（左の配列に挿入）

1	5	2	4	3	6
5	1	2	4	3	6
5	2	1	4	3	6
5	4	2	1	3	6
5	4	3	2	1	6
6	5	4	3	2	1

マージソート（大きい順にマージ）

分割

1	5	3	7	2	4	6	8
1	5	3	7	2	4	6	8
1	5	3	7	2	4	6	8
1	5	3	7	2	4	6	8

併合（マージ）

5	1	7	3	4	2	8	6
7	5	3	1	8	6	4	2
8	7	6	5	4	3	2	1

分けられなくなるまで分割
マージするときに大きい順に

クイックソート（ピボットで分類）

ピボット

4	2	0	7	3	1	6	5	8	9
7	6	5	8	9	4	2	0	3	1
7	6	5	8	9	4	2	0	3	1
8	9	7	6	5	4	3	2	0	1
8	9	7	6	5	4	3	2	0	1
9	8	7	6	5	4	3	2	1	0
9	8	7	6	5	4	3	2	1	0

ピボットより大きいデータ：左へ
ピボットより小さいデータ：右へ