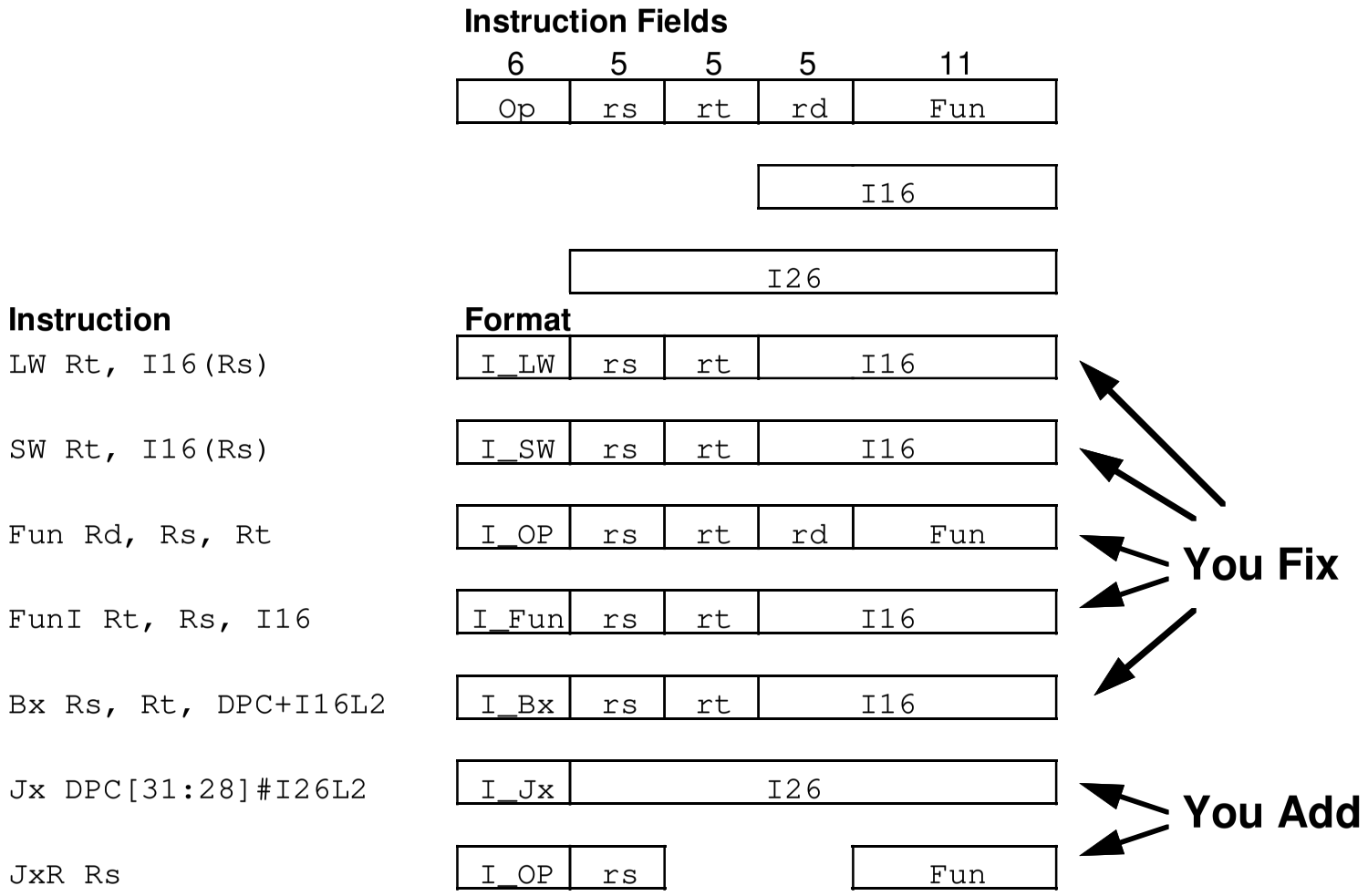
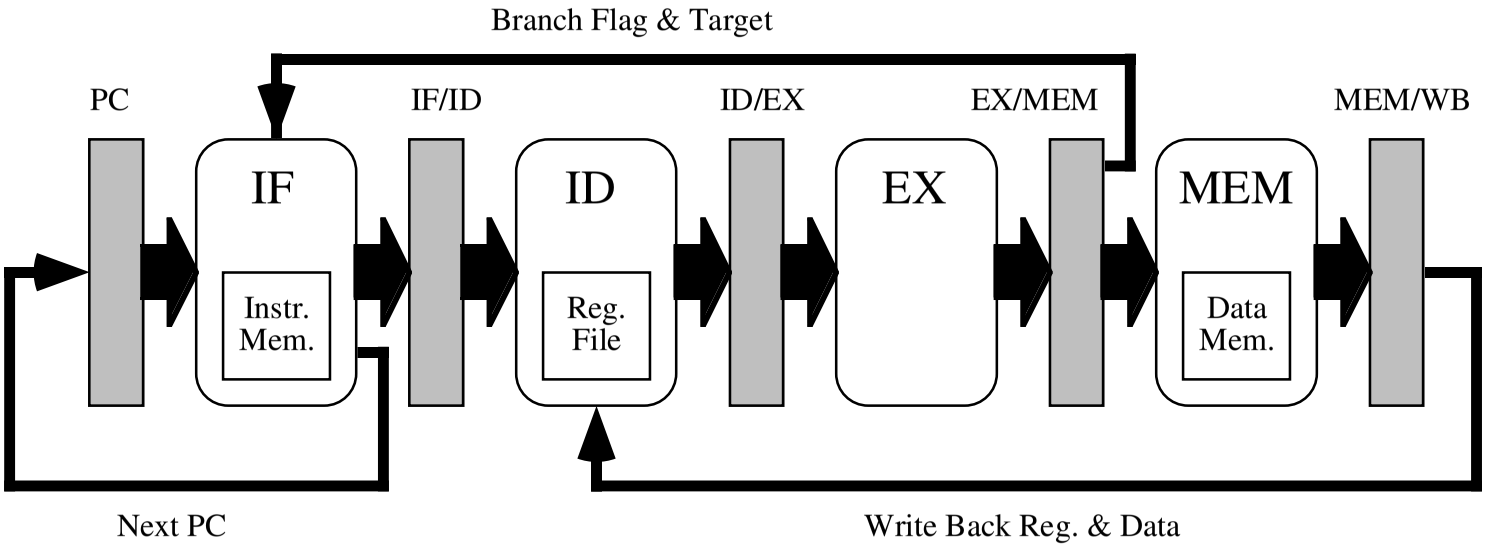
Notes on Assignment 2



pMIPS Pipeline Structure



Notes

- **Each stage consists of operate logic connecting pipe registers**
- **WB logic merged into ID**
- **Additional paths required for forwarding**

Source Code Files

Very Interesting

`ufix.c`	Only file you'll need to change

Somewhat Interesting

`mips.h`	Macro definitions of MIPS ISA
`simulate.h`	Interface to rest of simulation code
`sim.c`	Main simulation code
`pipe.h`	Interface to simulation utilities

Not Interesting

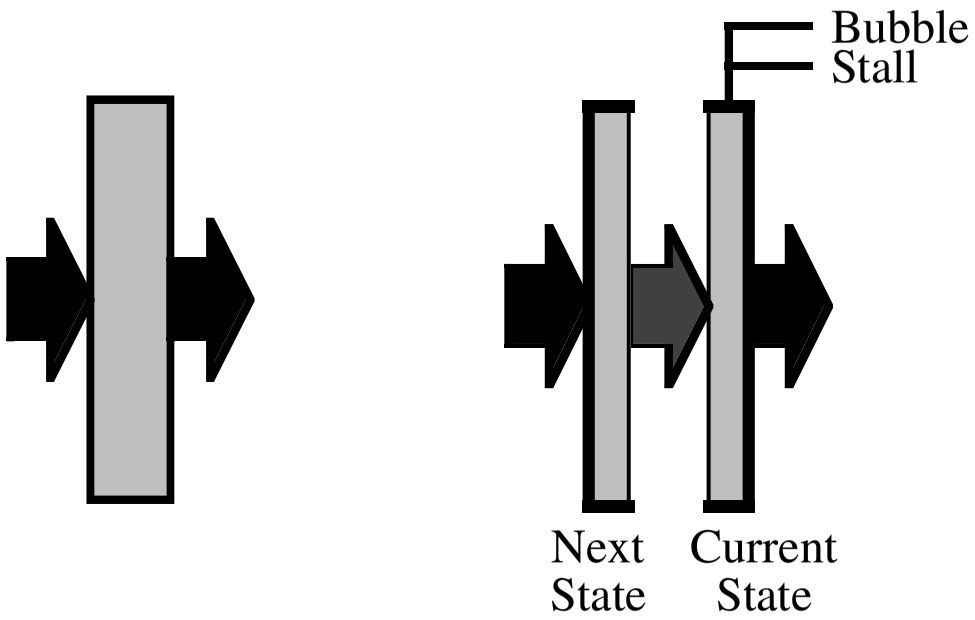
`pipe.c`	Simulation utilities
`tk_support.c`	Interface glue code
`mips_tk.tcl`	Graphic user interface
`tkAppInit.c`	Specific to Tk version (3.6)

Simulator Operation

```
/* sim.c */
/* Run pipeline for one cycle */
static x_status_t step_pipe()
{
  /* Update pipe registers */
  update_stages();
  /* Stage logic */
  do_if_stage();
  do_id_stage();
  do_ex_stage();
  do_mem_stage();
  /* Set pipe controls */
  do_stall_check(&pc_state->op, &if_id_state->op,
                 &id_ex_state->op, &ex_mem_state->op);
  /* Report progress */
  report_sys();
  return wb_in->exception;
}
```

- **Pipe registers updated**
  - `update_stages`
  - sets current states
- **Each stage implemented as separate procedure**
  - `do_`*`XX`*`_stage`
  - computes next states
- **Stall logic sets pipe register controls**
  - default is to update

Pipe Register



Operation

- **Pipe registers updated after next states computed**
  - Load next state into current
- **Special update options**
  - *Stall*: retain current state
  - *Bubble*: set current state to all 0's

Pipe Register Fields

Declaration

- **Define data type**
- **With appropriate fields**
- **Don't worry about tight bit packing**

```
/* simulate.h */
/* Program counter */
typedef struct {
  unsigned pc;
} pc_ele, *pc_ptr;
```

```
/* IF/ID pipe register */
typedef struct {
  unsigned instr;
  unsigned incr_pc;
  unsigned char exception;
  /* Debugging info */
  unsigned char stage_pc_valid;
  unsigned stage_pc;
} if_id_ele, *if_id_ptr;
```

```
pc_ptr pc_current, pc_next;
if_id_ptr if_out, id_in;
```

Pipe Register Allocation

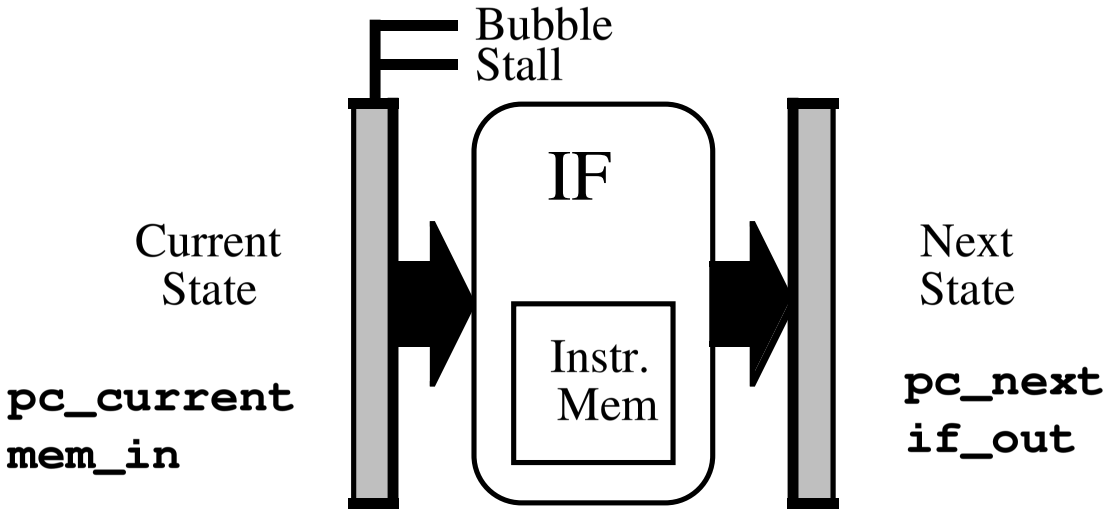
Creation

- **Request register of appropriate size**
- **Allocates storage for current and new states**

```
/* sim.c */
stage_ptr pc_state =
     new_stage(sizeof(pc_ele));
pc_next = pc_state->next;
pc_current = pc_state->current;
```

```
/* sim.c */
stage_ptr if_id_state =
    new_stage(sizeof(if_id_ele));
if_out = if_id_state->next;
id_in = if_id_state->current;
```

Pipeline Stage



Operation

- **Computes next state based on current**
  - From/to one or more pipe registers
- **May have embedded memory elements**
- **Signals encoded so that all 0's is like NOP**
  - When bubble generated

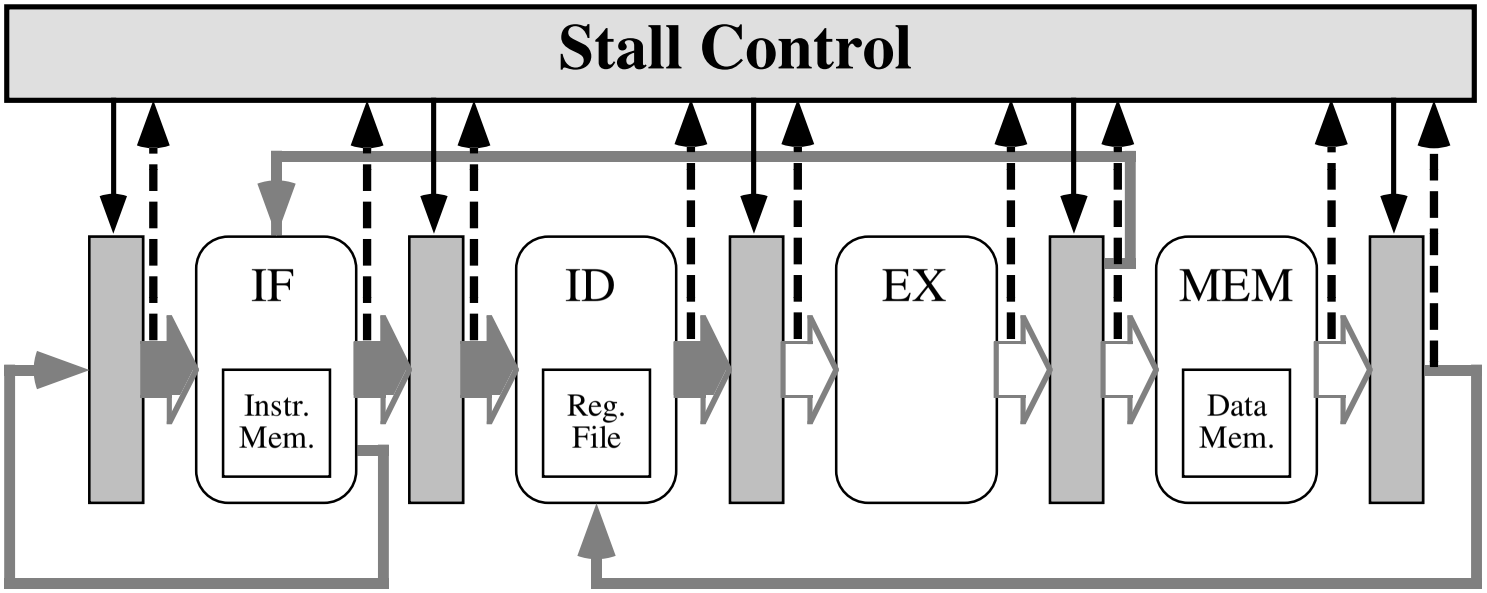
Stage Operation

```
void do_if_stage()
{
  unsigned pc = pc_current->pc;
  unsigned pc4 = pc+4;
  if_out->exception =
     instr_fetch(pc, &if_out->instr);
  if_out->incr_pc = pc4;
  pc_next->pc = mem_in->branch_flag ?
    mem_in->branch_target : pc4;
  if_out->stage_pc_valid =
    (if_out->exception == X_AOK);
  if_out->stage_pc = pc;
}
```

Procedure for Stage

- **Computes stage outputs based on inputs**
- **Embedded memories / ALU implemented as procedures**
- **Multiplexors, decoding logic, etc. implemented with C expressions**

Stalling



Stall Logic

- **Determines which stages to stall or bubble on next update**
- **Based on pipe register inputs and outputs**
  - Assumes stages have already operated

Stall Logic

```
void do_stall_check(p_status_t *pc_op, p_status_t *if_id_op,
                    p_status_t *id_ex_op, p_status_t *ex_mem_op)
{
  /* Should I stall the instruction in IF? */
  unsigned char stall_if = 0;
  /* Presently, No stalls.  */

  • • •

  else if (stall_if) {
    *pc_op = P_STALL;
    *if_id_op = P_BUBBLE;
```

Stall Check

- **Determines which registers to stall or bubble on next update**
- **Currently doesn't stall or bubble anything.**

Simulation Modes

Implementation

- **User Selects with Control Buttons**
- **Sets global variable `sim_mode`**
  - `S_WEDGED` Blunder through in current fashion
  - `S_STALL` Avoid hazards by stalling
  - `S_FORWARD` Avoid hazards by forwarding/cancelling