Arquitectura de Computadores
Grado de Informática

Implementación predictores

Departament d'Enginyeria Informàtica I Matemàtiques

Universitat Rovira i Virgili

Tarragona, Spain

Índice

Predictores de Salto

- **Simulador**

  sim-bpred, sim-outorder, etc

- **Implementados en los módulos:**

  bpred.h, bpred.h

- **Tipos**
  - Estáticos: nottaken, taken, perfect,
  - Dinámicos: bimod, 2lev(Gshare,Gselect:'Gag',Pag),comb

- **Funciones principales de los módulos:**
  - bpred_create(class,size)
  - bpred_lookup(pred,br_addr)
  - bpred_update(pred_addr,targ_addr,result)

Estructura

Main.c

```
main(){
```

sim-outorder.c

```
sim_reg_options()
...
sim_check_options()
...
sim_reg_stats()
...
sim_main() {
  For(;;){
      ruu_commit()
   ruu_writeback()
     lsq_refresh()
       ruu_issue()
    ruu_dispatch()
       ruu_fetch()
  }
}
```

bpred.c

```
bpred_create(){
  bpred_dir_create()
}

bpred_reg_stats(){
}

bpred_lookup(){
  bpred_dir_lookup()
  BTB
}

bpred_update(){
}
```

Índice

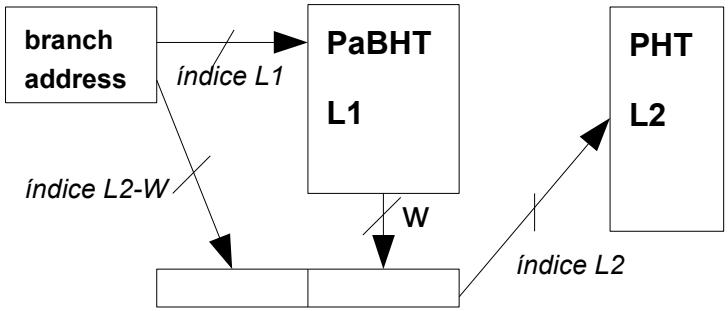
I. **Estructura general sim-outorder**

II. **Funciones a modificar: 2lev**

- Estructuras de datos
- Funciones en bpred.c
- Funciones en sim-outorder.c

Predictor 2lev

- **2lev implementa dos niveles de información:**
  - PaBHT: Historia de saltos por dirección de PC
  - PHT: Patrón de comportamiento



- **Se pueden simular varios predictores dentro del 2-level**

  -bpred:2lev <l1_size> <l2_size> <hist_size> <xor>

Figure 6. 2-level adaptive predictor structure

predictor	l1_size	hist_size	l2_size	xor
GAg	1	W	2^W	0
GAp	1	W	$>2^W$	0
PAg	N	W	2^W	0
PAp	N	W	2^{N+W}	0
gshare	1	W	2^W	1

Índice

I. **Estructura general sim-outorder**

II. **Funciones a modificar: 2lev**

- Estructuras de datos
- Funciones en bpred.c
- Funciones en sim-outorder.c

bpred.h

```
/* branch predictor types */
enum bpred_class {
  BPredComb,                    /* combined predictor (McFarling) */
  BPred2Level,                  /* 2-level correlating pred w/2-bit counters */
  BPred2bit,                    /* 2-bit saturating cntr pred (dir mapped) */
  BPredTaken,                   /* static predict taken */
  BPredNotTaken,                /* static predict not taken */
  BPred_NUM
};

/* direction predictor def */
struct bpred_dir_t {
  enum bpred_class class;       /* type of predictor */
  union {
    struct {
      unsigned int size;        /* number of entries in direct-mapped table */
      unsigned char *table;     /* prediction state table */
    } bimod;
    struct {
      int l1size;               /* level-1 size, number of history regs */
      int l2size;               /* level-2 size, number of pred states */
      int shift_width;          /* amount of history in level-1 shift regs */
      int xor;                  /* history xor address flag */
      int *shiftregs;           /* level-1 history table */
      unsigned char *l2table;   /* level-2 prediction state table */
    } two;
  } config;
};
```

Índice

I. **Estructura general sim-outorder**

II. **Funciones a modificar: 2lev**

- Estructuras de datos
- Funciones en bpred.c
- Funciones en sim-outorder.c

bpred_create

```
/* create a branch predictor */
struct bpred_t *                        /* branch predictory instance */
bpred_create(enum bpred_class class,    /* type of predictor to create */
             unsigned int bimod_size,   /* bimod table size */
             unsigned int l1size,       /* 2lev l1 table size */
             unsigned int l2size,       /* 2lev l2 table size */
             unsigned int meta_size,    /* meta table size */
             unsigned int shift_width,  /* history register width */
             unsigned int xor,          /* history xor address flag */
             unsigned int btb_sets,     /* number of sets in BTB */
             unsigned int btb_assoc,    /* BTB associativity */
             unsigned int retstack_size) /* num entries in ret-addr stack */
{
  struct bpred_t *pred;

  if (!(pred = calloc(1, sizeof(struct bpred_t))))
    fatal("out of virtual memory");

  pred->class = class;

  switch (class) {
  case BPred2Level:
    pred->dirpred.twolev =
      bpred_dir_create(class, l1size, l2size, shift_width, xor);

    BTB y RAS .....
}
```

bpred_dir_create

```
/* create a branch direction predictor */
struct bpred_dir_t *            /* branch direction predictor instance */
bpred_dir_create (
  enum bpred_class class,       /* type of predictor to create */
  unsigned int l1size,          /* level-1 table size */
  unsigned int l2size,          /* level-2 table size (if relevant) */
  unsigned int shift_width,     /* history register width */
  unsigned int xor)             /* history xor address flag */
{
  struct bpred_dir_t *pred_dir;
  unsigned int cnt;
  int flipflop;

  pred_dir = calloc(1, sizeof(struct bpred_dir_t))
  pred_dir->class = class;

  switch (class) {
  case Bpred2Level: {
      pred_dir->config.two.l1size = l1size;
      pred_dir->config.two.l2size = l2size;
      pred_dir->config.two.shift_width = shift_width;
      pred_dir->config.two.xor = xor;
      pred_dir->config.two.shiftregs = calloc(l1size, sizeof(int));
      pred_dir->config.two.l2table = calloc(l2size, sizeof(unsigned char));
      /* initialize counters to weakly this-or-that */
  ......}
  return pred_dir;
}
```

bpred_lookup

```
md_addr_t                                   /* predicted branch target addr */
bpred_lookup(struct bpred_t *pred,          /* branch predictor instance */
             md_addr_t baddr,               /* branch address */
             md_addr_t btarget,             /* branch target if taken */
             enum md_opcode op,             /* opcode of instruction */
             int is_call,                   /* non-zero if inst is fn call */
             int is_return,                 /* non-zero if inst is fn return */
             struct bpred_update_t *dir_update_ptr, /* pred state pointer */
             int *stack_recover_idx)        /* Non-speculative top-of-stack;
                                             * used on mispredict recovery */
{
  struct bpred_btb_ent_t *pbtb = NULL;
  int index, i;
  pred->lookups++;

  dir_update_ptr->dir.ras = FALSE;
  dir_update_ptr->pdir1 = NULL;
  dir_update_ptr->pdir2 = NULL;
  dir_update_ptr->pmeta = NULL;
  /* Except for jumps, get a pointer to direction-prediction bits */
  switch (pred->class) {
    case BPred2Level:
      if ((MD_OP_FLAGS(op) & (F_CTRL|F_UNCOND)) != (F_CTRL|F_UNCOND))
        {
          dir_update_ptr->pdir1 =
            bpred_dir_lookup (pred->dirpred.twolev, baddr);
        }
      Break;
```

..... RAS y BTB que ***obtiene la @ del salto***

bpred_dir_lookup

```
/* predicts a branch direction */
char *                                          /* pointer to counter */
bpred_dir_lookup(struct bpred_dir_t *pred_dir,  /* branch dir predictor inst */
                 md_addr_t baddr)               /* branch address */
{
  unsigned char *p = NULL;

  /* Except for jumps, get a pointer to direction-prediction bits */
  switch (pred_dir->class) {
    case BPred2Level:
      {
        int l1index, l2index;

        /* traverse 2-level tables */
        l1index = (baddr >> MD_BR_SHIFT) & (pred_dir->config.two.l1size - 1);
        l2index = pred_dir->config.two.shiftregs[l1index];
```

..... XOR o concatenacion

```
        l2index = l2index & (pred_dir->config.two.l2size - 1);

        /* get a pointer to prediction state information */
        p = &pred_dir->config.two.l2table[l2index];
      }
      Break;
  return (char *)p;
}
```

bpred_update

```
void
bpred_update(struct bpred_t *pred,      /* branch predictor instance */
             md_addr_t baddr,           /* branch address */
             md_addr_t btarget,         /* resolved branch target */
             int taken,                 /* non-zero if branch was taken */
             int pred_taken,            /* non-zero if branch was pred taken */
             int correct,               /* was earlier addr prediction ok? */
             enum md_opcode op,         /* opcode of instruction */
             struct bpred_update_t *dir_update_ptr)/* pred state pointer */
{
  struct bpred_btb_ent_t *pbtb = NULL;
  struct bpred_btb_ent_t *lruhead = NULL, *lruitem = NULL;
  int index, i;

  if (correct)
    pred->addr_hits++;

  if (!!pred_taken == !!taken)
    pred->dir_hits++;
  else
    pred->misses++;
..... RAS
```

bpred_update 2

```
..... RAS
  /* update L1 table if appropriate */
  /* L1 table is updated unconditionally for combining predictor too */
  if ((MD_OP_FLAGS(op) & (F_CTRL|F_UNCOND)) != (F_CTRL|F_UNCOND) &&
      (pred->class == BPred2Level || pred->class == BPredComb))
    {
      int l1index, shift_reg;

      /* also update appropriate L1 history register */
      l1index =
        (baddr >> MD_BR_SHIFT) & (pred->dirpred.twolev->config.two.l1size - 1);
      shift_reg =
        (pred->dirpred.twolev->config.two.shiftregs[l1index] << 1) | (!!taken);
      pred->dirpred.twolev->config.two.shiftregs[l1index] =
        shift_reg & ((1 << pred->dirpred.twolev->config.two.shift_width) - 1);
    }
.... BTB
  /* update state (but not for jumps) */
  if (dir_update_ptr->pdir1)
    {
      if (taken){
          if (*dir_update_ptr->pdir1 < 3)
            ++*dir_update_ptr->pdir1;
        }
      else { /* not taken */
          if (*dir_update_ptr->pdir1 > 0)
            --*dir_update_ptr->pdir1;
        }
    }
.... BTB
```

valor del contador saturado de PHT

Índice

I. **Estructura general sim-outorder**

II. **Funciones a modificar: 2lev**

- Estructuras de datos
- Funciones en bpred.c
- Funciones en sim-outorder.c

sim_reg_options

```
/* register simulator-specific options */
void
sim_reg_options(struct opt_odb_t *odb)
{
  opt_reg_int_list(odb, "-bpred:2lev",
                   "2-level predictor config "
                   "(<l1size> <l2size> <hist_size> <xor>)",
                   twolev_config, twolev_nelt, &twolev_nelt,
                   /* default */twolev_config,
                   /* print */TRUE, /* format */NULL, /* !accrue */FALSE);
```

sim_check_options

```
/* check simulator-specific option values */
void
sim_check_options(struct opt_odb_t *odb,        /* options database */
                  int argc, char **argv)        /* command line arguments */
{
  char name[128], c;

  if .......

  else if (!mystricmp(pred_type, "2lev"))
    {
      /* 2-level adaptive predictor, bpred_create() checks args */
      if (twolev_nelt != 4)
        fatal("bad 2-level pred config (<l1size> <l2size> <hist_size> <xor>)");
      if (btb_nelt != 2)
        fatal("bad btb config (<num_sets> <associativity>)");

      pred = bpred_create(BPred2Level,
                          /* bimod table size */0,
                          /* 2lev l1 size */twolev_config[0],
                          /* 2lev l2 size */twolev_config[1],
                          /* meta table size */0,
                          /* history reg size */twolev_config[2],
                          /* history xor address */twolev_config[3],
                          /* btb sets */btb_config[0],
                          /* btb assoc */btb_config[1],
                          /* ret-addr stack size */ras_size);
    }
```

Y ahora?


No empieces aquí
Empieza aquí!
Comienzo
Final
Donde Empezar
(tu presentación, artículo, libro, capítulo, etc.)