Paso 1)

Branch Prediction Accuracy (*not taken*): 40.2899%
IPC: 0.125866

(emacs o cualquier otro editor)
emacs branch/branch_predictor.cc

La predicción actual por defecto es:

NOT_TAKEN; // Always predict not taken

Paso 2)

Reemplazad la linea

```
return NOT_TAKEN; // Always predict not taken
```

por

```
return TAKEN; // Always predict taken
```

Ejecutad: `make clean; make`
y luego: `./run_champsim.sh traces/605.mcf.xz`

a) Branch Prediction Accuracy (*taken*): 59.7097%

b) Sí, el resultado tiene sentido, dado que es 100 menos la tasa de acierto de predecir NOT_TAKEN.

c) Dado que prediciendo “tomado” acertamos casi un 60% de las veces, esto nos indica que la mayoría de saltos se toman (un 60%).

d) **IPC: 0.177234**

¿Tendría sentido implementar predicción estática “Tomado” en DLX?
No, la dirección del salto no se sabe hasta el ciclo donde se conoce la condición del salto. Si no podemos adelantar la predicción no tiene sentido predecir sabiendo la dirección real.

e) ejecutamos `./run_champsim.sh traces/602.gcc.xz`

Branch Prediction Accuracy: 34.4666% para el predictor estático TAKEN
por lo tanto: 100 - 34.46 = 65,54% para el predictor estático NOT_TAKEN

Paso 3)

Código:

```
#include "ooo_cpu.h"
#define TAKEN true
#define NOT_TAKEN false
int contador_global;
void O3_CPU::initialize_branch_predictor()
{
  // Initialize your branch predictor here
    contador_global = 2;
}
uint8_t O3_CPU::predict_branch(uint64_t pc)
{
    if (contador_global >= 2)
            return TAKEN;
    else
            return NOT_TAKEN;
}
void O3_CPU::last_branch_result(uint64_t pc, uint8_t taken)
{
    // Taken contiene 0 o 1 en funcion de si se tomo o no
    if (taken) {
            if (contador_global < 3)
               contador_global++;
    } else {
            if (contador_global > 0)
               contador_global--;
    }
}
```

Resultados:

MCF
BP Accuracy: 39.8232% (antes 59.7097% para TAKEN y 40.2899% para NOT_TAKEN)
IPC: 0.171377

GCC
BP Accuracy: 65.596% (antes 34.4666% para TAKEN y 65,54 para NOT_TAKEN)
IPC: 0.0956109

¿Por qué no es mejor el predictor dinámico?

La aplicación puede tener múltiples saltos, y cada uno puede tener un comportamiento opuesto. Al tener un contador global compartido, el contador se decrementa e incrementa arbitrariamente, y no proporciona información útil para todos los saltos. Esto es lo que se conoce como ALIASING, donde varios saltos mapean al mismo registro.

¿Cuándo funcionaría? → Únicamente si todos los saltos se comportasen igual (algo muy raro).

Paso 4)

```
#include "ooo_cpu.h"
#define TAKEN true
#define NOT_TAKEN false
#define NUM_CONT 64

int contador_global[NUM_CONT];

void O3_CPU::initialize_branch_predictor()
{
  // Initialize your branch predictor here
    for (int i = 0; i < NUM_CONT ; i++)
    contador_global[i] = 2;
}

uint8_t O3_CPU::predict_branch(uint64_t pc)
{
    int indice = pc % NUM_CONT;
    if (contador_global[indice] >= 2)
            return TAKEN;
    else
            return NOT_TAKEN;
}

void O3_CPU::last_branch_result(uint64_t pc, uint8_t taken)
{
    int indice = pc % NUM_CONT;
    // Taken contiene 0 o 1 en funcion de si se tomo o no
    if (taken) {
            if (contador_global[indice] < 3)
               contador_global[indice]++;
    } else {
            if (contador_global[indice] > 0)
               contador_global[indice]--;
    }
}
```

Resultados:

MCF
BP Accuracy: 99.9556%
IPC: 0.172146

GCC
BP Accuracy: 98.863%
IPC: 0.120913

Tamaño de la tabla de predicción

64 entradas a 2 bits por entrada = 128 bits

Paso 5)

Tamaño	lbm	wrf
64	94.918%	85.0038%
128	94.918%	92.0997%
256	94.918%	93.7457%
512	94.918%	93.9203%
1024	94.918%	93.9285%

wrf obtiene mayor beneficio.
lbm no se ve afectado.

64 entradas para lbm.
256 o 512 entradas para wrf.

El ALIASING afecta a wrf pero no a lbm, ya que no se ve afectada cuando aumentamos el número de entradas. Cuando deja de mejorar es que prácticamente no queda ALIASING, o que los saltos que tienen ALIASING se comportan igual.

Paso 6)

```
#include "ooo_cpu.h"

#define TAKEN true
#define NOT_TAKEN false

#define NUM_CONT 1024

int contador_global[NUM_CONT];
int historia_global;

void O3_CPU::initialize_branch_predictor()
{
  // Initialize your branch predictor here
    for (int i = 0; i < NUM_CONT ; i++) {
            contador_global[i] = 2;
    }
    historia_global = 0;
}

uint8_t O3_CPU::predict_branch(uint64_t pc)
{
    int indice = (pc ^ historia_global) % NUM_CONT;

    if (contador_global[indice] >= 2)
            return TAKEN;
    else
            return NOT_TAKEN;
}

void O3_CPU::last_branch_result(uint64_t pc, uint8_t taken)
{
    int indice = (pc ^ historia_global) % NUM_CONT;
    // Taken contiene 0 o 1 en funcion de si se tomo o no
    if (taken) {
            if (contador_global[indice] < 3)
               contador_global[indice]++;
    } else {
            if (contador_global[indice] > 0)
               contador_global[indice]--;
    }

    historia_global = historia_global << 1; // Desplazo un bit a la izq, inserta un 0
    if (taken) historia_global++; // sumo 1 si fue tomado
}
```

lbm BP Accuracy: 96.6566%

wrf BP Accuracy: 94.8463%

Paso 7)

Mejora = ((grande / pequeño) - 1)*100

IPCs	GCC	MCF	LBM	WRF	**M. Armónica**
IPC–Gshare	0.121783	0.172162	0.367326	0.55629	0.21575
IPC–Not Taken	0.0956788	0.125866	0.36137	0.441029	0.17071
Mejora	27,283 %	36,781 %	1,648 %	26,134 %	26.38392

Media armónica (ver Tabla).