Architetture Avanzate

Domande campione appello

Tempo a disposizione: 2h 30min

Parte 1 – teoria (20 punti)

1. (8 punti) Si elenchino i vantaggi e gli svantaggi tra memoria condivisa e memoria distribuita in architetture parallele.
2. (8 punti) Si consideri la riscrittura di un codice sequenziale con l'obiettivo di trasformarlo in parallelo per poter essere eseguito su una macchina dotata di 100 processori basata su paradigma MPI. L'obiettivo della riscrittura è di raggiungere uno speedup di 80x. Qual è la frazione massima del programma che, teoricamente, può rimanere sequenziale?
3. (10 punti) Il profiling di una applicazione eseguita su un calcolatore evidenzia che il 25% delle istruzioni sono FP, il CPI medio di ogni istruzione FP è 4.0, il CPI medio delle altre istruzioni è 1.33. Tra le istruzioni FP, la radice quadrata viene eseguita con una frequenza del 2% e il CPI di tale istruzione è 20.0. Tra le seguenti alternative:
    a. Investire in logica per diminuire il CPI della radice quadrata FP a 2.0
    b. Investire in logica per diminuire il CPI di tutte le istruzioni FP a 2.5

    Qual è porta maggiori benefici in termini di performance?

4. (10 punti) Si descrivano le 4 macro fasi da implementare durante la parallelizzazione di un programma.
5. (10 punti) Si descrivano 3 tecniche di branch prediction statiche (implementate dal compilatore).
6. (4 punti) Nel codice seguente, si identifichino gli hazard di tipo data (data dependency), classificandole in true data dependency o false (name) dependency:

```
L.D F0,0(R1); F0=array element
ADD.D F4,F0,F2; add scalar of F2
S.D F4,0(R1); store result
DADDUI R1,R1,#-8; decr pointer 8 B
BNE R1,R2,Loop; branch R1!=R2
```

7. (14 punti) Si assuma un'architettura MIPS (presentata durante il corso) con una pipeline a 5 stadi. Si considerino gli stalli dovuti alle dipendenze tra istruzioni come da tabella seguente:

Instruction producing result	Instruction using result	Latency in clock cycles
FP ALU op	Another FP ALU op	3
FP ALU op	Store double	2
Load double	FP ALU op	1
Load double	Store double	0

Qual è l'incremento di performance del seguente codice applicando loop unrolling e scheduling in temrini di cicli di clock risparmiati per iterazione del ciclo?:

```
L.D F0,0(R1); F0=array element
ADD.D F4,F0,F2; add scalar of F2
S.D F4,0(R1); store result
DADDUI R1,R1,#-8; decr pointer 8 B
BNE R1,R2,Loop; branch R1!=R2
```

8. (12 punti) Si descriva l'approccio Tomasulo per l'implementazione della tecnica di esecuzione out-of-ordering delle istruzioni.
9. (10 punti) Si descriva il concetto di Hardware-based speculation.
10. (15 punti) Si considerino i seguenti misses per 1000 istruzioni di cache I-data+D-data vs. cache unificata, ottenuti tramite l'esecuzione di un SW che causa accessi in memoria pari a 74% per istruzioni e 26% per dati:

Cache size	I-cache	D-cache	Unified
8KB	8.16	44.9	63.0
16KB	3.82	40.9	51.0
32KB	1.36	38.4	43.3
64KB	1.61	36.9	39.4
128KB	0.30	35.3	36.2
256KB	0.02	32.6	32.9

Quale configurazione di cache (16K-I + 16K-D o unificata) garantisce il più basso miss rate assumendo che il 36% di istruzioni sono istruzioni di trasferimento dati (data write/read)?

Si assuma che uno hit richiede 1cc e che il miss penality è 100 cc. Uno hit di un'istruzione load o store richiede 1 cc extra nella cache unificata. Qual è il tempo di accesso in memoria medio (AMAT) in entrambi i casi? Si assumano le cache di tipo write through con write buffer e si ignorino gli stalli dovuti al write buffer.

11. (15 punti) Si assuma una cache con miss penalty pari a 200 cc e che tutte le istruzioni richiedono 1 cc. Si assuma un miss rate medio è pari al 2%, una media di accessi in memoria per istruzione pari a 1.5 e un numero di cache misses per 1000 istruzioni pari a 30. Qual è l'impatto sulle performance quando la cache è attiva rispetto a quando non lo è?
12. (10 punti) Si descrivano le tecniche conosciute per la riduzione del miss penalty in un calcolatore con cache.
13. (15 punti) Si illustri e commenti la macchina a stati finiti implementata nei blocchi di una cache per la gestione della coerenza della cache con protocollo di snoopy assumendo una politica write-back del dato.

Parte 2 - Lab (10 punti)

1. (3 punti) Si vuole usare ogni thread di una GPU per calcolare 8 elementi di una somma tra vettori. Ogni blocco di thread processa 8*blockDim.x elementi consecutivi che formano 8 sezioni. Tutte le thread di ogni blocco prima processano una sezione, dove ogni thread processa un elemento. Poi tutti assieme si spostano sulla prossima sezione, dove ogni thread processerà un elemento. Si assuma che la variabile i è l'indice del primo elemento da processare da una thread. Quale dovrebbe essere l'espressione per mappare gli indici delle thread/blocchi sugli indici dei dati del primo elemento?

   a) i = blockIdx.x * blockDim.x + threadIdx.x + 8
   b) i = (blockIdx.x * blockDim.x + threadIdx.x) * 8
   c) i = blockIdx.x * blockDim.x * 8
   d) i = blockIdx.x * blockDim.x * 8 + threadIdx.x

2. (3 punti) A che livello è condivisa la shared memory in un device GPU e a che livello lavora la primitiva `__synchhreads()`:
   a. Blocco ; warp
   b. Warp ; warp
   c. Blocco ; blocco
   d. SM ; blocco
   e. Nessuna delle risposte sopra
3. (3 punti) Per una somma tra vettori, si assuma che i vettori siano di lunghezza 9000, che ogni thread calcola 9 elementi di output, e che ogni blocco sia dimensionato a 256 thread. Il programmatore deve configurare il kernel in modo da avere il minimo numero di blocchi per coprire tutti gli elementi. Quante thread verranno create nella griglia?
   a. 1000
   b. 900
   c. 1024
   d. 2000
   e. 2048
   f. 8292
4. (3 punti) Se un device GPU ha Streaming Multiprocessor (SM) che possono tenere fino a 1536 thread e massimo 8 blocchi, quale delle seguenti configurazioni porterà al maggior numero di thread per SM?
   a. 64 thread per blocco
   b. 128 thread per blocco
   c. 256 thread per blocco
   d. 1024 thread per blocco

5. (3 punti) In una moltiplicazione tra matrici implementata in CUDA utilizzando shared memory e tile da 16x16, qual è la riduzione dell'uso di memory bandwidth per matrici M e N WIDTHxWIDTH?
6. (5 punti) Si descriva, con una primitiva esempio, il concetto di collective comunication in MPI.
7. (8 punti) Si illustri la differenza delle tecniche di scheduling per loop in OpenMP (`#pragma omp for schedule`)

8. (8 punti) Partendo dal seguente codice, si implementi la moltiplicazione tra matrici con utilizzo di shared memory, considerando

```
__global__
void matrixMult_ShM(int* d_M, int* d_N, int Width, int* d_P)
{

}
```