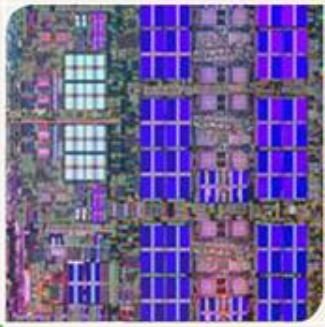
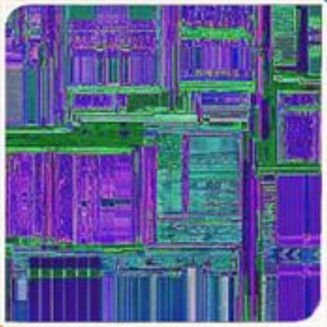



CS4803DGC Design Game Console

Spring 2010

Prof. Hyesoon Kim

Georgia Tech | College of Computing

Review for Quiz-III

ARM processor

Nintendo DS programming

(ARM Assembly coding)

- Interrupt/polling method
- Immediate operands, shift
- Memory indexing
- Memory mapped I/O processors
- Condition code
- Predicated execution
- Fixed point operations
- SPI connections
- Basic graphics (translation, scaling, rotation)
- Split transactions
- Advanced Microcontroller Bus Architecture (AMBA)
- Handling nested function calls

Review topics..

- LRB architecture: U&V pipe
  - Ring network
  - Gather/scatter operations
- Cell processors
  - Heterogeneous architecture
  - In-order/out-of-order processors
  - DMA, Load/store managements
  - Branch handling
  - Power efficient design

Review topics..

- MIPS
  - Delayed branch
  - Pipeline stalls
- Performance questions

Amdahl's Law

- Speedup when only fraction of program can be parallelized

$$\text{Speedup} = \frac{\text{Execution Time without Enhancement}}{\text{Execution Time with Enhancement}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}}$$

$$\text{Execution Time}_{\text{new}} = \text{Execution Time}_{\text{old}} \times \left(\left(1 - \text{Fraction}_{\text{Enhanced}}\right) + \frac{\text{Fraction}_{\text{Enhanced}}}{\text{Speedup}_{\text{Enhanced}}} \right)$$

$$\text{Overall Speedup} = \frac{1}{\left(\left(1 - \text{Fraction}_{\text{Enhanced}}\right) + \frac{\text{Fraction}_{\text{Enhanced}}}{\text{Speedup}_{\text{Enhanced}}} \right)}$$

$\text{Speedup}_{\text{Enhanced}} = 20$ $\text{Fraction}_{\text{Enhanced}} = 0.1$ VS $\text{Speedup}_{\text{Enhanced}} = 1.2$ $\text{Fraction}_{\text{Enhanced}} = 0.9$

$$\text{Speedup} = \frac{1}{\left((1 - 0.1) + \frac{0.1}{20} \right)} = 1.105$$

$$\text{Speedup} = \frac{1}{\left((1 - 0.9) + \frac{0.9}{1.2} \right)} = 1.176$$

Questions related to Amdahl's Law

Func A 80% , Func B 20%

- After parallelizing Func A, we run the parallelized program on an 128-core GPU machine. What is the maximum speedup?

$$\text{Overall Speedup} = \frac{1}{\left((1 - \text{Fraction}_{\text{Enhanced}}) + \frac{\text{Fraction}_{\text{Enhanced}}}{\text{Speedup}_{\text{Enhanced}}}\right)}$$

$$\text{Speedup} = \frac{1}{\left((1 - 0.8) + \frac{0.8}{128}\right)} = 4.85$$

Questions related to Amdahl's Law-II

Func A 80% , Func B 20%

- We run this code in a cell. Parallelized func A is run on 7 SPEs and non-parallelized Fun B is run on 1 PPE. PPE is 80% faster than base line and the speed of SPE is the same as baseline. What is the overall speedup?

$$\text{Speedup} = \frac{\text{Execution Time without Enhancement}}{\text{Execution Time with Enhancement}} = \frac{\text{Execution Time}_{\text{old}}}{\text{Execution Time}_{\text{new}}}$$

$$\text{Execution Time}_{\text{new}} = \text{Execution Time}_{\text{old}} \times \left(\left(1 - \text{Fraction}_{\text{Enhanced}}\right) + \frac{\text{Fraction}_{\text{Enhanced}}}{\text{Speedup}_{\text{Enhanced}}} \right)$$

$$\text{Execution Time}_{\text{new}} = \text{Execution Time}_{\text{old}} \times \left(\frac{0.8_{\text{Enhanced}}}{7_{\text{Enhanced}}} + \frac{0.2_{\text{Enhanced}}}{1.8_{\text{Enhanced}}} \right)$$

$$\text{Speedup} = \frac{1}{\left(\frac{0.8}{7} + \frac{0.2}{1.8} \right)} = 4.43$$

Thumb ISA

- The native ARM ISA has a 32-bit instruction format. To reduce the instruction code size, ARM introduced the Thumb ISA. The Thumb ISA has a 16-bit instruction format but not all native instructions are translated into one Thumb instruction. The following table shows how the ARM Native instructions are translated into the Thumb ISA.

Native ISA	Thumb ISA
ADD	ADD
MAD	ADD, MUL
BRCMP	CMP, BR
BR	BR

Native ISA	static	dynamic
ADD	5	50
MAD	2	10
BRCMP	1	5
BR	2	10

- Q: I-cache size is 256B, I-cache miss rate for Native ISA vs. Thumb ISA?
- I-cache miss execution latency is 10 cycles, all other instructions takes 1 cycle. Total execution time difference?

ARM Immediate operands

- Using 12 bits how to represent 32-bit immediate value?
- Immediate = (0→255) x 2^{2n}
  - Where 0≤n≥ 12 (4 bits rotation)
  - 8 bit immediate + 4-bit shift
  - 8 bit + 24 = 32 bit representation
- Which values can be represented by immediate values?
  - 0x4300
  - 0x4592
  - 0x5600
  - 0x360120

ARM Block Memory Operations

0x207	7
0x206	6
0x205	5
0x204	4
0x203	3
0x202	2
0x201	1
0x200	0

MEM addr / Mem data

r10 = 0x204
LDMIA r10, {r0,r1,r4}

What's the value of r0, r1, r4?

STMIB r10, {r0,r1,r4}

The contents of memory?

Condition codes

```
MOV r1 #1
ADDS r0, r2, r3
MOVHI r1 #0
```

- What would be r1 value if r2 = 0x8000 r3=0x3fff

- We will provide the condition code table.

0.8.8 fixed Point Conversion

Fixed point number: 0x500

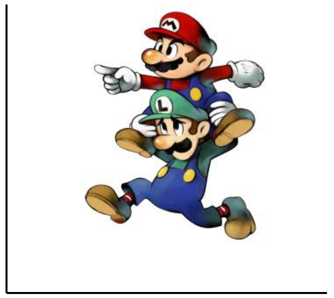
- Value ?
- 5

Fixed point number: 0x104

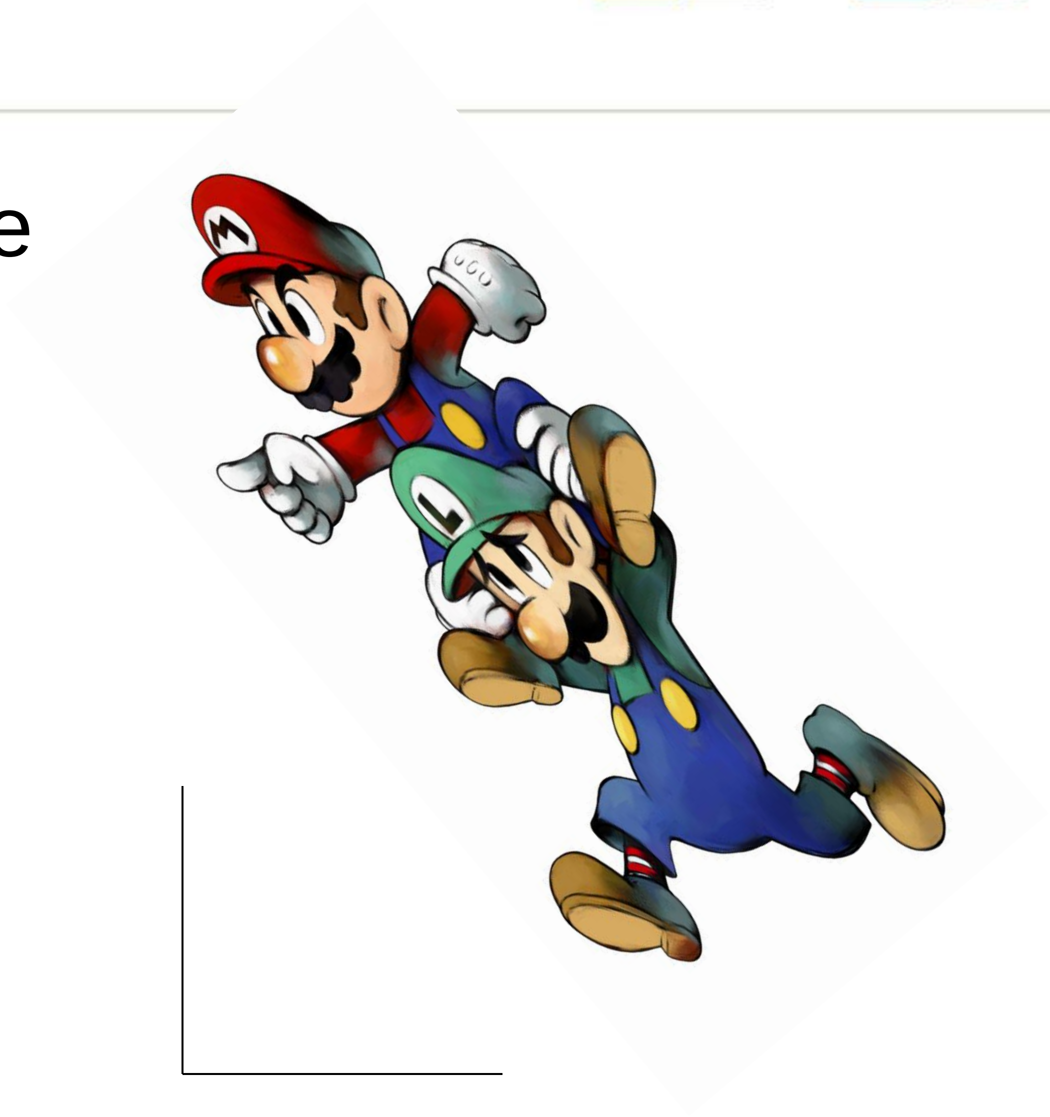
- Value?
- 0x104=b1 0000 0100 = 1 + 0*1/2+0*1/4+0*1/8+0*1/16+0*1/32+1*1/64+0*1/128+0*/256
- Value 4.75 → Fixed point?
- 4.75 = 4 + 0.5+ 0.25 → (4 << 8 | 1 << 7 | 1 << 6)
- = 0x4C0

Graphics 101

- Find the affine transform matrix



Original image
(1,1) (1,2), (2,1) ,(2,2)



Transformed images

(8,1) (10,3), (1,11) ,(6,13)

Delayed Branch

```
              sub r1, r2,r3
0x800         add r4, r2,r3
0x804         br      target
0x808
0x80b
0x810
0x900 target mul r2,  r3,r4
```

```
0x900 target mul r2,  r3,r4
```

- This is a code w/o delayed branch slots. MIPS uses 2 instruction delayed slots. What will be the new code when there is 2 instruction delayed slots?

Announcements

- Wed (4/28)
  - 5 min for team presentation + demo (nintendo Ds)
  - Bring your notebook and Nintendo DS kit
  - Peer review (review itself is also graded)
- F (4/30)
  - Final project submission
  - Report, code
- M (5/3)
  - 11:30 AM 1:30 exam, bring your calculator
  - Return Nintendo DS kit (no return, no update in your grade.)