

Sample Problems for Midterm Exam

CDA4101

1.4. Assume a color display using 8 bits for each of the primary colors (red, green, blue) per pixel and a frame size of 1280 x 1024.

- a) What is the minimum size in bytes of the frame buffer to store a frame?

$$3 \times 12 \quad 3932160 \text{ bytes}$$

- b) How long would it take, at a minimum, for the frame to be sent over a 100 Mbit/s network?

$$\frac{3932160 \times 8}{100 \times 10^6} = \frac{31,457,280}{100,000,000} = 0.3145728 \text{ s}$$

1.5. Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

- a) Which processor has the highest performance expressed in instructions per second?

$$P1 = \frac{3,000,000,000 \text{ Hz}}{1.5 \text{ CPI}} \quad P2 = \frac{2,500,000,000 \text{ Hz}}{1 \text{ CPI}} \quad P3 = \frac{4,000,000,000 \text{ Hz}}{2.2 \text{ CPI}}$$

$$P1 = 2,000,000,000 \text{ Ins} \quad P2 = 2,500,000,000 \text{ Ins} \quad P3 = 1,818,181,818 \text{ Ins.}$$

- b) If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.

$$P1 = 20,000,000,000 \text{ Ins} \quad P2 = 25,000,000,000 \quad P3 \text{ Cycles} = 10 \times 4,000,000,000$$

$$P1 \text{ Cycles} = 20,000,000,000 \times 1.5 \quad P2 \text{ Cycles} = 25,000,000,000 \quad P3 \text{ Cycles} = 40,000,000,000$$

$$P1 \text{ Cycles} = 30,000,000,000 \quad P3 = 1.818 \times 10^9$$

- c) We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

$$0.7t = \frac{N \times 1.2 \text{ CPI}}{x \text{ CR}} \quad 0.7 \left(\frac{N \times \text{CPI}}{\text{CR}} \right) = \frac{N \times 1.2 \text{ CPI}}{x \text{ CR}}$$

$$0.7 = \frac{1.2}{x} \quad x = \frac{1.2}{0.7}$$

$$0.7x = 1.2 \quad x = 1.714286$$

1.6. Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (class A, B, C, and D). P1 with a clock rate of 2.5 GHz and CPIs of 1, 2, 3, and 3, and P2 with a clock rate of 3 GHz and CPIs of 2, 2, 2, and 2.

Given a program with a dynamic instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which implementation is faster?

$$\begin{array}{l}
 \text{P1: } \\
 \text{CPIs: } \text{A: } 1, \text{B: } 2, \text{C: } 3, \text{D: } 3 \\
 \text{CR: } 2.5 \times 10^9 \text{ Hz} \\
 \text{Ins: } 1,000,000 \\
 \text{P: } 100,000 \times 1 = 100,000 \\
 \text{T: } \frac{100,000 \times 1}{2.5 \times 10^9} = 4 \times 10^{-6} \text{ s} \\
 \text{T_A: } \frac{100,000 \times 1}{100,000} = 1 \\
 \text{T_B: } \frac{200,000 \times 2}{1,000,000} = 0.4 \\
 \text{T_C: } \frac{500,000 \times 3}{1,000,000} = 1.5 \\
 \text{T_D: } \frac{200,000 \times 3}{1,000,000} = 0.6 \\
 \text{Total: } 4 \times 10^{-6} \text{ s}
 \end{array}$$

$$\begin{array}{l}
 \text{P2: } \\
 \text{CPIs: } \text{A: } 2, \text{B: } 2, \text{C: } 2, \text{D: } 2 \\
 \text{CR: } 3 \times 10^9 \text{ Hz} \\
 \text{Ins: } 1,000,000 \\
 \text{P: } 100,000 \times 2 = 200,000 \\
 \text{T: } \frac{200,000 \times 2}{3 \times 10^9} = 0.0006666666666666667 \text{ s} \\
 \text{T_A: } \frac{100,000 \times 2}{100,000} = 0.0002 \\
 \text{T_B: } \frac{200,000 \times 2}{1,000,000} = 0.0004 \\
 \text{T_C: } \frac{500,000 \times 2}{1,000,000} = 0.001 \\
 \text{T_D: } \frac{200,000 \times 2}{1,000,000} = 0.0004 \\
 \text{Total: } 0.0006666666666666667 \text{ s}
 \end{array}$$

- a) What is the global CPI for each implementation?

$$P1_{GCPi} = 2.6 \quad P2_{GCPi} = 2$$

- b) Find the clock cycles required in both cases.

1,000,000 ins.

$$2.6 \times 10^6 \quad 2 \times 10^6$$

2.1. For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, and i are given and could be considered 32-bit integers as declared in a C program. Use a minimal number of MIPS assembly instructions.

$$\text{Add: } 10, h - 5 \quad f = g + (h - 5);$$

Add f, g, & h

2.2. For the following MIPS assembly instructions above, what is a corresponding C statement?

$f = i + (g+h)j$ add f, g, h
 add f, i, f

2.3. For the following C statement, what is the corresponding MIPS assembly code? Assume that the variables f, g, h, i, and j are assigned to registers \$s0, \$s1, \$s2, \$s3, and \$s4, respectively. Assume that the base address of the arrays A and B are in registers \$s6 and \$s7, respectively.

Sub \$t0,
Sll \$t0,\$t0,
lw \$t0

B[8] = A[i - j]; sub \$t0, \$s3, \$s4
 sll \$t0, \$t0, 2
 add \$t1, \$s6, \$t0
 lw \$t2, 0(\$t1)
 sw \$t2, 32(\$s7)

2.4. For the MIPS assembly instructions below, what is the corresponding C statement? Assume that the variables f, g, h, i, and j are assigned to registers \$s0, \$s1, \$s2, \$s3, and \$s4, respectively. Assume that the base address of the arrays A and B are in registers \$s6 and \$s7, respectively.

```

sll $t0, $s0, 2    # $t0 = f * 4
add $t0, $s6, $t0  # $t0 = &A[f]
sll $t1, $s1, 2    # $t1 = g * 4
add $t1, $s7, $t1  # $t1 = &B[g]
lw  $s0, 0($t0)    # f = A[f]
addi $t2, $t0, 4   # $t2 = &A[f+1]
lw  $t0, 0($t2)    # $t0 = A[f+1]
add $t0, $t0, $s0  #$t0 = A[f]+A[f+1]
sw  $t0, 0($t1)    # B[g] = A[f]+A[f+1]

```

$$B[g] = A[f] + A[f+1];$$

2.7. Show how the value 0xabcdef12 would be arranged in memory of a little-endian and a big-endian machine. Assume the data is stored starting at address 0.

Address	Big	Little
6	ab	12
1	cd	ef
2	ef	cd
3	12	ab

2.8. Translate 0xabcdef12 into decimal.

$$(16^7 + (1 \times 16^6) + (15 \times 16^5) + (4 \times 16^4) + (13 \times 16^3) + (12 \times 16^2) + (11 \times 16^1) + (10 \times 16^0))$$

2.9. Translate the following C code to MIPS. Assume that the variables f, g, h, i, and j are assigned to registers \$s0, \$s1, \$s2, \$s3, and \$s4, respectively. Assume that the base address of

the arrays A and B are in registers \$s6 and \$s7, respectively. Assume that the elements of the arrays A and B are 4-byte words:

```
shl $t0, $s3, 2          B[8] = A[i] + A[j];  
shl $t1, $s4, 2  
add $t0, $t0, $s6  
add $t1, $t1, $s6  
lw $t2, 0($t0)  
lw $t3, 0($t1)  
add $t4, $t2, $t3  
sw $t4, 32($s7)
```

2.26 Consider the following MIPS loop:

```
LOOP: slt $t2, $0, $t1 0 < 10 True fl False = 0  
      beq $t2, $0, DONE  
      subi $t1, $t1, 1  
      addi $s2, $s2, 2  
      j LOOP
```

DONE:

2.26.1. Assume that the register \$t1 is initialized to the value 10. What is the value in register \$s2 assuming \$s2 is initially zero? $\frac{2}{2} \times 10 = 20$

2.26.2. For each of the loops above, write the equivalent C code routine. Assume that the registers \$s1, \$s2, \$t1, and \$t2 are integers A, B, i, and temp, respectively.

```
while (i > 0) {  
    i--;  
    B += 2i;  
}
```

2.26.3. For the loops written in MIPS assembly above, assume that the register \$t1 is initialized to the value N. How many MIPS instructions are executed?

$\frac{5}{2}n + 2$

2.27. Translate the following C code to MIPS assembly code. Use a minimum number of instructions. Assume that the values of a, b, i, and j are in registers \$s0, \$s1, \$t0, and \$t1, respectively. Also, assume that register \$s2 holds the base address of the array D.

```
for (i = 0; i < a; i++)
```

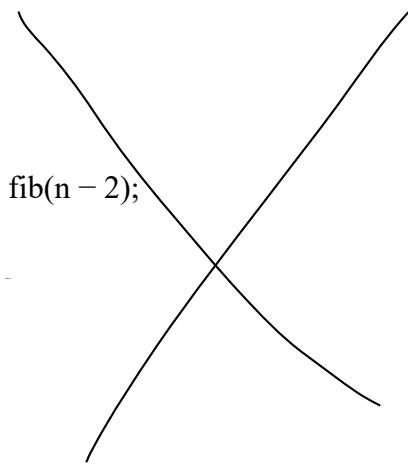
```
for(j = 0; j < b; j++)
    D[4 * j] = i + j;
```

```

add $t0, $0, $0 #i=0
L1 slt $t2, $t0, $30 #Loop con. L3: add,$t0,$t0,1
    beq $t2, $0, DONE
    add $t1, $0, $0
L2 slt $t3, $t1, $5
    beq $t3, $0, L3
    add $t4, $t0, $t1 #i+j
    sll $t5, "
    add $t5, $t1 } 2
    sw , $t1, -($t0)
    add: $t1, $t1, 1
j L2
DONE:
```

2.31. Implement the following C code in MIPS assembly. What is the total number of MIPS instructions needed to execute the function?

```
int fib(int n){
    if (n == 0)
        return 0;
    else if (n == 1)
        return 1;
    else
        return fib(n - 1) + fib(n - 2);
}
```



3.1. What is 5ED4–07A4 when these values represent unsigned 16-bit hexadecimal numbers? The result should be written in hexadecimal. Show your work.

$$\begin{array}{r}
 5ED4 \\
 -07A4 \\
 \hline
 5730
 \end{array}
 \quad
 \begin{array}{r}
 0101\ 1110\ 1101\ 0100 \\
 -0000\ 0111\ 1010\ 0100 \\
 \hline
 0101\ 0111\ 0011\ 0000
 \end{array}$$

3.2. What is 5ED4–07A4 when these values represent signed 16-bit hexadecimal numbers stored in sign-magnitude format? The result should be written in hexadecimal. Show your work.

$$\begin{array}{r}
 5ED4 \\
 -07A4 \\
 \hline
 5730
 \end{array}
 \quad
 \begin{array}{r}
 0101\ 1110\ 1101\ 0100 \\
 +1111\ 1000\ 0101\ 1100 \\
 \hline
 0101\ 0111\ 0011\ 0000
 \end{array}$$

3.4. What is 4365–3412 when these values represent unsigned 12-bit octal numbers? The result should be written in octal. Show your work.

$$\begin{array}{r}
 4365 \\
 -3412 \\
 \hline
 0753
 \end{array}
 \quad
 \begin{array}{r}
 100\ 011\ 110\ 101 \\
 -011\ 100\ 001\ 010 \\
 \hline
 000\ 111\ 101\ 011
 \end{array}$$

3.8. Assume 185 and 122 are signed 8-bit decimal integers stored in sign-magnitude format. Calculate 185–122. Is there overflow, underflow, or neither?

$$\begin{array}{r}
 185 \\
 -2 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 185 \\
 -12 \\
 \hline
 63
 \end{array}
 \quad
 \begin{array}{r}
 185 \\
 -122 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 1011\ 1001 \\
 +1000\ 0110 \\
 \hline
 0011\ 1111
 \end{array}
 \quad
 \text{Underflow}$$

3.9. [10] <§3.2> Assume 151 and 214 are signed 8-bit decimal integers stored in two's complement format. Calculate 151+214 using saturating arithmetic. The result should be written in decimal. Show your work.

$$\begin{array}{r}
 100\ 01 \\
 214 \\
 \hline
 121
 \end{array}
 \quad
 \begin{array}{r}
 110\ +0010\ 1010 \\
 \hline
 1111\ 0011
 \end{array}
 \quad
 \begin{array}{r}
 0110\ 1001 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 \curvearrowleft 1111111
 \end{array}$$

3.23. Write down the binary representation of the decimal number 63.25 assuming the IEEE 754 single precision format.

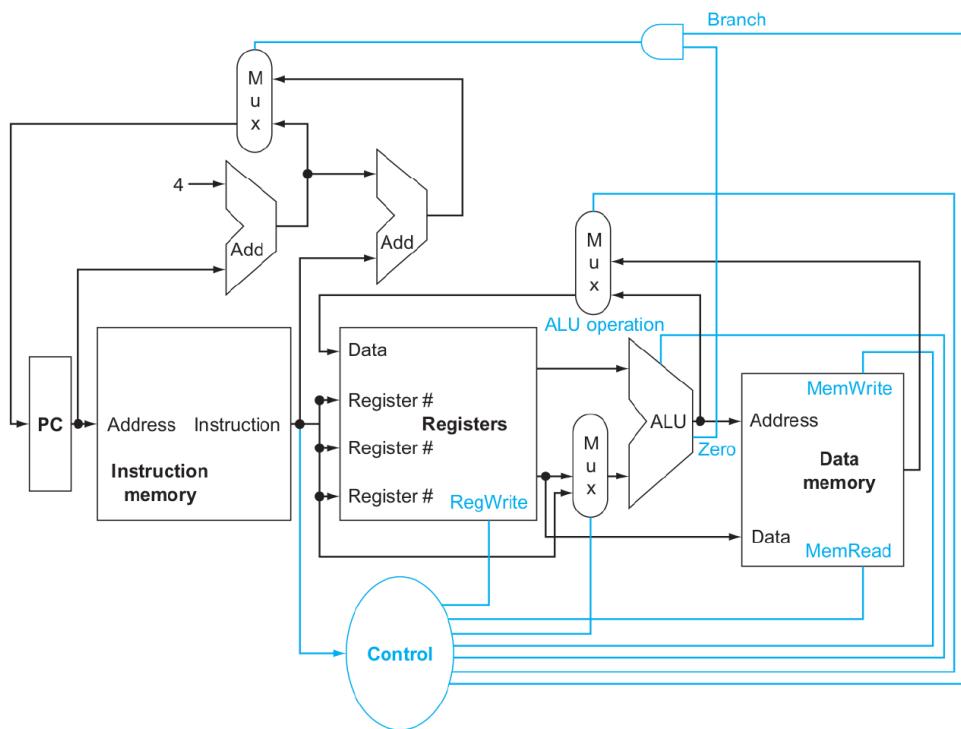
$$63.25 \times 4 = \frac{253}{4} = \frac{1111101}{2^2} = 11111.01 = 1.11101 \times 2^5 \quad | 2^{7+5}=132$$

01000010011110100000000000000000

4.2 The basic single-cycle MIPS implementation in COD Figure 4.2 (The basic implementation of the MIPS subset ...) can only implement some instructions. New instructions can be added to an existing Instruction Set Architecture (ISA), but the decision whether or not to do that depends, among other things, on the cost and complexity the proposed addition introduces into the processor datapath and control. The first three problems in this exercise refer to the new instruction:

Instruction: LWI Rt, Rd(Rs)

Interpretation: $\text{Reg[Rt]} = \text{Mem}[\text{Reg[Rd]} + \text{Reg[Rs]}]$



4.2.1. Which existing blocks (if any) can be used for this instruction?

4.2.2. Which new functional blocks (if any) do we need for this instruction?

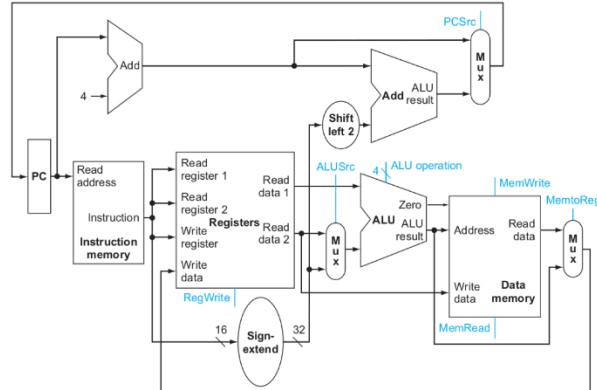
4.2.3. What new signals do we need (if any) from the control unit to support this instruction?

4.4 Problems in this exercise assume that logic blocks needed to implement a processor's datapath have the following latencies:

I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-Left-2
200ps	70ps	20ps	90ps	90ps	250ps	15ps	10ps

4.4.1. If the only thing we need to do in a processor is fetch consecutive instructions (COD Figure 4.6 (A portion of the datapath used for fetching instructions and incrementing the program counter)), what would the cycle time be?

4.4.2. Consider a datapath similar to the one in COD Figure 4.11 (The simple datapath for the core MIPS architecture ...), but for a processor that only has one type of instruction: unconditional PC-relative branch. What would the cycle time be for this datapath?



4.4.3. Repeat 4.4.2, but this time we need to support only conditional PC-relative branches.

4.8 In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

alu	beq	lw	sw
45%	20%	20%	15%

4.8.1. What is the clock cycle time in a pipelined and non-pipelined processor?

4.8.2. What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

4.8.3. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

4.8.4. Assuming there are no stalls or hazards, what is the utilization of the data memory?

4.8.5. Assuming there are no stalls or hazards, what is the utilization of the write-register port of the "Registers" unit?

Extra questions: Explain two ways to resolve data hazard in pipelining. Compare them. What are the pros/cons for each one.