CSE 410 - Spring 2004

Homework 4

due on Friday, April 30 at 9:30 AM, at the beginning of class

25 points

Name Solution DWJ

Student # ____________

1a. (2pt) Fill in the "usage", "available?" and "restore required" columns in the following chart.

Usage?	Brief description of the conventional usage of the register.
Available?	Is the register available for use in user code, or is it reserved?
Restore?	If the register is available for use, does a procedure have to restore the value of the register after using it? (If not available for use, put n/a.)

name	*available?*	*restore?*	*usage?*
zero	yes	no	read-only, always returns 0
at	no	n/a	reserved for use as assembler temporary
v0,v1	yes	no	return results from procedure
a0-a3	yes	no	pass 1st four arguments to procedure
t0-t9	yes	no	for use as needed (no restore)
s0-s7	yes	yes	for use as needed (restore)
fp (s8)	yes	yes	frame pointer or general usage (restore)
sp	yes	yes	stack pointer
ra	yes	yes	return address
gp	yes	yes	pointer to global data area
k0,k1	no	n/a	for use by kernel (operating system)

1b. (2pt) The registers listed above are remarkably general purpose, compared to designs of earlier systems. A different set of conventions could assign most of the functions to different registers without changing the underlying MIPS hardware. However, this isn't true of all the registers. Identify two registers listed above for which a change in usage would require a change in the hardware. Describe why you selected these two registers.

zero - value cannot be changed - always 0

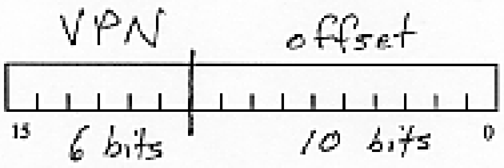
ra - the jal instruction writes the return address in ~~$~~ ra

2. Consider a machine that has a 16-bit program address space (logical address space). This is considerably smaller than the systems we have been studying.

a. (2pt) What is the largest logical address that a program in this system can use? Give your answer in both decimal base 10 and hexadecimal base 16 notation.

65535_{10} $FFFF_{16}$

b. (2pt) The designers have implemented a Virtual Memory system that uses 1KB pages, so the page offset field is 10 bits wide (2^{10}=1024). Using the drawing of a 16-bit logical address word given below, indicate the Virtual Page Number Field and the offset field.



c. (2pt) How many Virtual Page Numbers are there in this system? Give your answer in 2^n notation, using the correct number for "n".

$2^6 = 64$

d. (2pt) If the physical addresses are 24 bits wide, how many Physical Page Numbers are there? Give your answer in 2^n notation, using the correct number for "n".

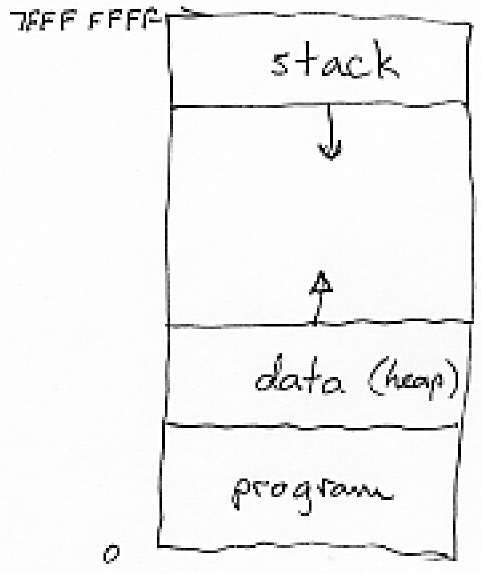
$2^{14} = 16384$

e. (2pt) How much physical memory is required to implement a system with 24-bit physical addresses? Give your answer in MegaBytes, where 1 MB = 1048576 bytes.

$1\,MB = 1048576 = 2^{20}$

$$\frac{2^{24}\ \text{bytes}}{2^{20}\ \text{bytes/MB}} = 2^4\ MB = 16\ MB$$

3. (2pt) Draw a simple picture showing a program address space as we have studied it indicating where the program code, the data (the heap), and the stack are located. Show the directions that the heap and the stack grow while the program is executing.



More detail is shown on the lecture slides, but this is all the question asked for.

4a. (2pt) Describe the general characteristics of a program that would exhibit high temporal locality of data references but low spatial locality of data references.

Frequent references to widely separated data.

Example: A program that repeatedly calculated the position of a point moving in space under the influence of several forces. All the values would be used frequently, but ~~[illegible]~~ they could be located at arbitrary, unrelated locations in memory.

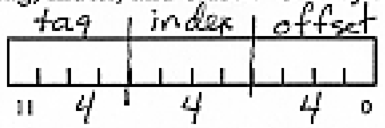
4b. (2pt) Describe the general characteristics of a program that would exhibit high temporal locality of instruction fetches and high spatial locality of instruction fetches.

Frequent references to instructions located near each other.

This is the classic use of a cache. Any short instruction loop that fits entirely within the instruction cache is an example. Counting bytes in a zero-terminated string is a small example. Generally, a program that performs a limited number of functions implemented in code that is located in a small neighborhood of addresses satisfies this condition.

5. (5pt) The cache shown in this problem is direct mapped, and there are 16 4-word blocks in the cache. The cache is initially empty. Memory is byte-addressed using 12-bit addresses.

a. Show how you will divide up a 12-bit address into the required fields for accessing the cache: tag, index, and offset. Clearly identify how many bits are allocated to each field.



b. Consider the sequence of data memory references to the right. Fill in the cache contents by considering each memory reference in turn. Show if it's a hit or a miss, then fill in the data in the proper places below. You may need to overwrite some entries as you proceed.

Sequence of memory references

Referenced Address$_{16}$	Hit or Miss?	Address Contents$_{16}$
2A1	Miss	F0
B03	Miss	F1
133	Miss	F2
134	Hit	F3
1A0	Miss	F4
CCC	Miss	F5
210	Miss	F6

Cache contents

Index$_{16}$	Tag	Valid?	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
			Data (draw a line across a cell if data is valid but not given in the problem statement)															
0	B	Y	—	—	—	F1	—	—	—	—	—	—	—	—	—	—	—	—
1	2	Y	F6	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
2	—	N																
3	1	Y	—	—	—	F2	F3	—	—	—	—	—	—	—	—	—	—	—
4	—	N																
5	—	N																
6	—	N																
7	—	N																
8	—	N																
9	—	N																
A	~~2~~ 1	Y	F4	~~F0~~	—	—	—	—	—	—	—	—	—	—	—	—	—	—
B	—	N																
C	C	Y	—	—	—	—	—	—	—	—	—	—	—	—	F5	—	—	—
D	—	N																
E	—	N																
F	—	N																