

CSC258 Computer Organization 2011 Fall

Assignment 3 due Thu.Dec.1 at 6pm in BA2220 and electronically

1[35] There is a wonderful website called “Peter Answers” at peteranswers.com. It seems to be able to answer any question, even a personal question, like “What are you looking at?”. To use it, you need to know a secret about how it works. With the secret, you can amaze naïve people who do not know the secret. Write a program in CSC258 assembly language (no indexing, no indirection) that works like the website. Here is an example output when used by someone who knows the secret.

```
Petition: Peter, please a:
Question: Who is prime minister?
Answer: Stephen Harper
```

The computer types `Petition: .` The knowledgeable user types `.Stephen Harper:` but this is not output to the screen; instead the computer types `Peter, please a: .` The initial period tells the computer that this is a knowledgeable user, and that the user is typing the answer, and that the computer should type as much of `Peter, please answer the following question if you can:` as necessary to disguise what the knowledgeable user is typing. The knowledgeable user explains to the naïve victim that we don't need the whole petition. It is the colon that tells the computer that the user is finished typing the answer. Then the computer types `Question:` and the user types `Who is prime minister?`; the question mark tells the computer that the user is finished typing the question. The computer types `Answer: Stephen Harper .` Here is an example output when used by someone who does not know the secret.

```
Petition: Peter, please answer:
Question: Who is prime minister?
Answer: I do not answer a nonbeliever.
```

The computer types `Petition: .` The naïve user types `Peter, please answer: .` The lack of an initial period tells the computer that the user is naïve, and the computer should echo the input up to and including the colon. Then the computer types `Question:` and the user types `Who is prime minister?` and the computer types `Answer: I do not answer a nonbeliever.` The execution should be repeated until the petition is the escape character. There is a routine named `printstr` available on the course website that you may find useful. Suggestion: write the program in the language of your choice, and then translate to assembly language. You can assemble and run your program using `ax.c` from the course website and from `cdf`. You can compile and run `ax` on any computer, but you should make a final test on `cdf` because that's where it will be marked.

What happens if the answer is longer than `Peter, please answer the following question if you can?`

You must hand in the program and commentary and a sample output on paper in the box in BA2220. You must also submit your program electronically, either by using the `submit` command on `cdf`, or by using `www.cdf.utoronto.ca/students` and click on `csc258` and then on `submissions`. The assignment name is `A3Q1` and the file must be named `A3Q1.ax`. The deadline is firm; submissions are not accepted after that. (Suggestion: for safety, submit something well in advance, but keep working and resubmit.) The program should be lightly but judiciously commented, and the first line should say your name and student number. The paper commentary on the design should be less than one page.

(question 2 is on the next page)

- 2 Write a microprogram (sequence of register transfers) for the unoptimized CSC258 computer that implements each of the following instructions. Describe any special details or changes to the basic computer that are needed to add these instructions to the machine. Do not make any changes to the machine that are not needed.
- (a)[5] BII m (Branch Indexed Indirect) Branch to the memory location whose address is contained in memory location $m + (\text{contents of AC})$. The accumulator and memory remain unchanged, but the E bit may change. This instruction is useful for a “switch” or “case” statement.
- (b)[5] CSK m (Compare and SKip) The contents of AC are compared to the contents of memory location m (as 2's complement integers). If AC contains the smaller number, execution continues with the next instruction. If the two are equal, the next instruction is skipped. If AC contains the greater number, the next two instructions are skipped. The accumulator and memory remain unchanged, but the E bit may change.