CS 245 Spring 2013
Assignment 3 – Parsing an I-formatted instruction

For all assignments, be sure to:

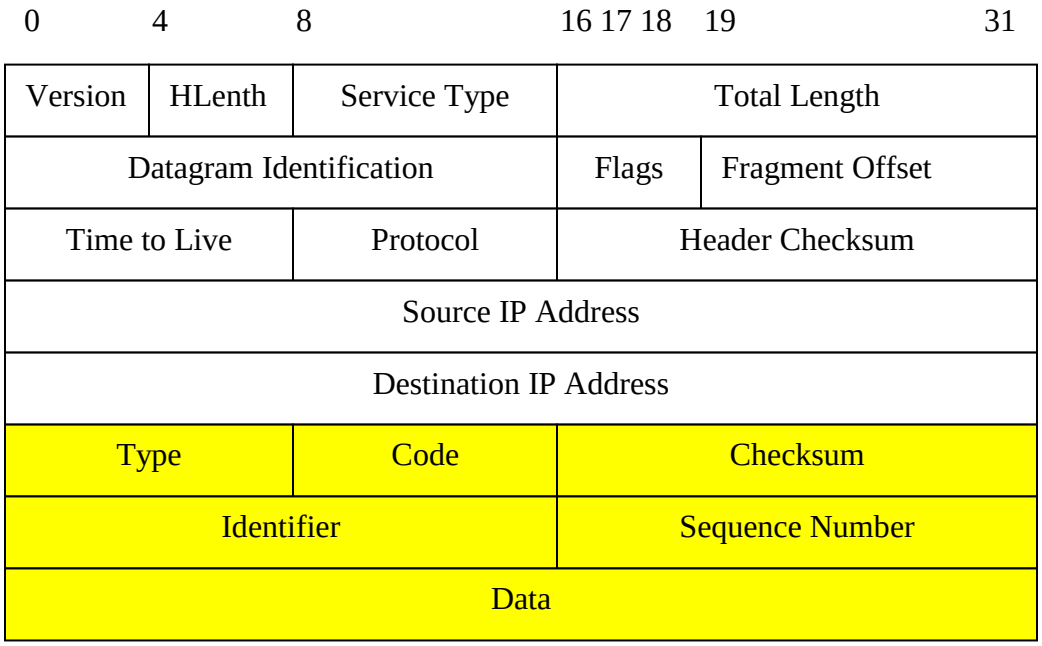
- Include file comments at the top, listing your name, the name of the program, and briefly describe what the program does.
- Describe the register convention, and calling sequence for procedures.
- Line up instruction parts in columns: 0=labels; 1tab=instruction mnemonic; 2tabs=operands; 3-4tabs=comments.
- Include pseudo-code comments to the right of your assembly code. **Avoid** comments that tell me what the assembly does, such as: move 25 to register $s3 – I already know this. Keep your comments to the logic.

Program 3: ***Parsing part of an Internet Protocol (IP) Header***

You may remember this exercise from the Binary-Hexadecimal handout.
In this assignment, you will parse an IP header to print certain important fields in the header.
Your output should look as follows (not showing accurate data for below input):

IPv:4 Length:1000 Datagram ID: 12345 More: 1 TTL: 128 Protocol: 6
Source IP: 196.25.33.23 Dest IP: 193.42.66.77

You may be interested in understanding what each field means. At the bottom of this document I have a description of what these most important fields are used for. Below is a diagram showing how the IP header is formatted, and where the different fields are found. Note the bit numbers at the top. The More Fragment flag is bit 18 in word 2. A better diagram can found at Wikipedia: http://en.wikipedia.org/wiki/IPv4#Header



Data that you may use as input (hard-coded) is the hexadecimal data:

4500 05dc 039c 2000 8001 902b c0a8 0004
c0a8 0005 0800 2859 0200 1c00 6162 6364
6566 6768 696a 6b6c 6d6e 6f70 7172 7374

Coding Requirements

It is important for you to implement this with assembly procedures. In your main(), parse each field, then send the name and value to print to PrntFld(). To print the IP addresses, from main() call IPaddr() which is your implementation for homework assignment 2. You should implement the following procedures as shown below. Take these comment headers directly for use with your code.

Part 1: Passing arguments to procedures using $a registers.

```
#################################################################
#   Print Field:  PrntFld
#   This procedure prints a name and its decimal value, for example:  Name: 25
#
#   Arguments:  $a0 = Name to print
#               $a1 = Value to print
#
#   Stack Usage:  0($sp) = Return Address
#################################################################
#################################################################
#   Print IP Address:  IPaddr
#   This procedure parses an IP version 4 address and prints it.
#   The routine calls PrntFld to actually print each part of the IP address.
#
#   Arguments:  $a0 = Name to print (Source IP or Dest IP)
#               $a1 = Full IP address
#
#   Stack Usage:  0($sp) = Return Address
#################################################################
```

Part 2: Passing arguments to procedures on the stack

```
#################################################################
#   Print Field:  PrntFld
#   This procedure prints a name and its decimal value, for example:  Name: 25
#
#   Arguments:  -4($sp) = Name to print
#               -8($sp) = Value to print
#
#   Stack Usage:  8($sp) = Name to print
#                 4($sp) = Value to print
#                 0($sp) = Return Address
#################################################################
#################################################################
#   Print IP Address:  IPaddr
```

```
#   This procedure parses an IP version 4 address and prints it.
#   The routine calls PrntFld to actually print each part of the IP address.
#
#   Arguments:  -4($sp) = Name to print (Source IP or Dest IP)
#               -8($sp) = Full IP address
#
#   Stack Usage:  <You define>
##############################################################
```

Have IPaddr() call PrntFld(). PrntFld can print the periods and the decimal numbers. To get this to work, you will need to save off the return address $ra on the stack in IPaddr(), so you can return to your main().

Submission

Turn this program in as assign3.asm **via paper and electronic copy**:

```
$ submit 245 assign3a.asm assign3b.asm
```

Grading

Each homework assignment is worth 10 points. Be careful to include all comments and format correctly, as directed in the beginning of this file.

Description of IP header

The Internet Protocol (IP) is part of the TCP/IP protocol that is used in the Internet. IP is responsible for routing; explicitly getting a packet from the source node to the destination node. That is why the IP packet header has both a source IP address (sender) and destination IP address (receiver).

The IP version will either be 4 or 6. It is the version of the IP protocol. The header we have been working with is IP version 4, which uses smaller IP address sizes: 32 bits. The Internet Protocol became so popular that the whole world (literally) now wants to use it. So IP version 6 uses 4 words for each source and destination IP address.

The length is the length of the packet. We are only seeing the IP header, but it must carry data too.

The Datagram ID and More Fragment bit (bit 18) are used if a packet is split up, because it is too long for some network that the packet traverses. The router may split up long packets and send them as shorter packets. The Destination must put all the packet pieces back together again. The Datagram ID tells the destination what packet number this partial packet fits into – it helps the destination put the packet-puzzle back together again. The More bit indicates if there are more partial packets that go into the larger packet or not. The last packet always has a More bit of 0 (False).

The Time To Live (TTL) is a counter that makes sure that a packet does not do an infinite loop forever in a network due to errors in routing tables. Each router that receives the packet decrements the TTL, and when the TTL becomes zero, the packet is deleted.

The Protocol field defines what the next protocol header is. Most packets will contain TCP headers too, and potentially HTTP or SMTP (email) protocol headers. Alternatively the protocol header may be UDP, ICMP (for pings and error messages), Domain Name Server or other protocols.