ECE473 Quiz 3 Oct 19, 2008

Name: KEY

1.[10] In the most basic and general sense, what does the utility "make" do?
- If the target is a file......

It checks for dependencies on the file + if they are newer than the target the target is recreated using the commands in the rule

- If the target is a "phony target" with no dependencies....

the commands in the rule are executed

2.[8] In the rule below, label: target, prerequisite, command, where the tab goes.

target prerequisite command

```
$(PRG).hex: $(PRG).elf
<tab> $(OBJCOPY) -j .text -j .data -O ihex $< $@
```

3.[8] When an interrupt occurs on the Mega128, just before vectoring to the ISR, the CPU hardware clears the global interrupt bit (GIE), and the program counter is loaded with the address of the corresponding vector which is usually a jump instruction. After the jump is taken and the ISR entered, at least two "housekeeping chores" need to be taken care of. What are they? Be specific.

Saving any used registers
Saving status register

4.[4] The last instruction of an interrupt service routine is Reti and its function is to set the global interrupt enable again, (+ pop PC back from stack)

5.[4] What does the "volatile" modifier indicate to the compiler?

that the variable may be changed outside the present scope so it may not be optimized away.

7.[4] The article "Debugging Embedded C", lists the four phases in the process of debugging. Name two of them.

testing localization
stabilization correction

OR

make the bug repeatable make corrections
isolate the problem retest

*

8.[30] For the makefile below, write out the commands that would be executed. Do this in the order of execution. Assume that quiz_code.c has initially been compiled by typing "make all" and that no compile errors occurred. List all the "switches" to the commands that are executed.

a.[10] "make all"
b.[10] "make program"
c.[10] "make clean"

KEY

```
# $@ file name of target, i.e., left hand side of :
# $< name of first prerequisite
# $? name of all the prerequisites that are newer than the target
# $^, $+ names of all the prerequisites

PRG             = quiz_code
OBJ             = $(PRG).o
MCU_TARGET      = atmega128
OPTIMIZE        = -O2    # options are 1, 2, 3, s
CC              = avr-gcc
OBJCOPY         = avr-objcopy
OBJDUMP         = avr-objdump

override CFLAGS         = -g -Wall $(OPTIMIZE) -mmcu=$(MCU_TARGET)
override LDFLAGS        = -Wl,-Map,$(PRG).map

all:    $(PRG).elf lst

$(PRG).elf: $(OBJ)
        $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $^

program: $(PRG).hex
        avrdude -c usbasp -p m128 -e -U flash:w:$(PRG).hex  -v

$(PRG).hex: $(PRG).elf
        $(OBJCOPY) -j .text -j .data -O ihex $< $@

lst:    $(PRG).lst

%.lst:  %.elf
        $(OBJDUMP) -h -S $< > $@

→ clean:
        rm -rf *.o $(PRG).elf
        rm -rf *.lst *.map

#==========================================================================
```

a. [10] "make all"

if the source file was compiled initially, "make All" does nothing.
Make: Nothing to be done for 'All'.

b. [10] "make program"

→ Avr-objcopy -j .text -j .data -O ihex quiz-code.elf quiz-code.hex
→ Avrdude -c usbasp -p m128 -e -U Flash:w: quiz-code.hex -v

c. [10] "make clean"

rm -rf *.o quiz-code.elf
rm -rf *.lst *.map