# Final Exam S3 <br> Computer Architecture 

Duration: 1 hr 30 min
Write answers only on the answer sheet.

## Exercise 1 (4 points)

Complete the table shown on the answer sheet. Write down the new values of the registers (except the PC) and memory that are modified by the instructions. Use the hexadecimal representation. Memory and registers are reset to their initial values for each instruction.

Initial values:


## Exercise 2 (3 points)

Complete the table shown on the answer sheet. Give the result of the additions and the values of the $\mathbf{N}, \mathbf{Z}$, $\mathbf{V}$ and $\mathbf{C}$ flags.

## Exercise 3 (4 points)

Let us consider the following program. Complete the table shown on the answer sheet.

```
Main move.l #$5A9500,d7
next1 moveq.l #1,d1
    tst.b d7
    beq next2
    moveq.l #2,d1
next2 moveq.l #1,d2
    cmpi.w #$95,d7
    bgt next3
    moveq.l #2,d2
next3 clr.l d3
    move.l #$512,d0
loop3 addq.l #1,d3
    subq.b #1,d0
    bne loop3
next4 clr.l d4
    move.l #64,d0
loop4 addq.l #1,d4
    dbra d0,loop4 ; DBRA = DBF
quit illegal
```


## Exercise 4 ( 9 points)

All the questions in this exercise are independent. Except for the output registers, none of the data or address registers must be modified when the subroutine returns. For the whole exercise, a pair of ASCII characters is made up of two characters representing an integer between 0 and 99 (in base 10). For instance, the characters " 08 " (' 0 ' and ' 8 ') represent the value 8 ; and the characters " 42 " (' 4 ' and ' 2 ') represent the value 42 . As a reminder, the ASCII code of the ' 0 ' character is equal to $\$ 30$.

## Be careful. All the subroutines must contain 15 lines of instructions at the most.

1. Write the ToAscii subroutine that converts an integer into a pair of ASCII characters.

Input: D0.L holds an integer between 0 and 99 (in base 10).
A0.L points to a 2-byte buffer where the pair of ASCII characters must be written.

Example: If D0 $=\$ 0000002 \mathrm{~A}$ (42 in base 10) and A0.L $=\$ 00005000$, then the address $\$ 5000$ will hold the value $\$ 34$ (ASCII code of ' 4 ') and the address $\$ 5001$ will hold the value $\$ 32$ (ASCII code of ' 2 ').
2. Write the ToInt subroutine that converts a pair of ASCII characters into an integer.

Input: A0.L points to a pair of ASCII characters to convert.
Output: D0.B holds the integer that is represented by the pair of ASCII characters.
Be careful, only D0.B must be modified (bits from 8 to 31 must not be modified).

Example:

| Main | movea.l \#ascii,a0 <br> jsr <br> illegal ToInt |
| :--- | :--- | :--- |
| ascii | dc.b "42" |$\quad ; D 0 . B=\$ 2 A$ (42 in base 10)

3. By using the ToInt subroutine, write the GetSum subroutine that returns the sum of two pairs of ASCII characters.
Input: A0.L points to a first pair of ASCII characters.
A1.L points to a second pair of ASCII characters.
Output: D0.B holds an integer that is the sum of the two pairs of ASCII characters.
Be careful, only D0.B must be modified (bits from 8 to 31 must not be modified).

Example:

| Main | movea.l \#ascii1, a0 <br> movea.l \#ascii2, a1 <br> jsr GetSum <br> illegal | $; D 0 . B=\$ 32$ (50 in base 10) |
| :--- | :--- | :--- |
|  | dc.b "42"  <br> ascii1  <br> ascii2 dc.b$\quad$ "08" |  |

Computer Architecture－EPITA－S3－2020／2021
EASy68K Quick Reference v1．8 http：／／www．wowgwep．com／EASy68K．htm Copyright © 2004－2007 By：Chuck Kelly

| Dpcade | Siza | Iperand | CLR |  | Effect | tive A | Addres | s s | ource， | dest | ， | ， | ， | lacement |  | Dparation | Descriptian |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | BWL | s，d | XNZVC | Dn | An | （An） | （An）＋ | －（An） | （i，An） | （i，An，Rn） | abs．W | abs．L | （i．，PC） | （i，$, C, R \mathrm{Rn}$ ） | \＃n |  |  |
| ABCD | $B$ | $\begin{aligned} & D y, D x \\ & -(A y)-(A x) \end{aligned}$ | ＊U＊${ }^{\text {＊}}$ | E |  |  | － | 8 | － | － | － | － | － | － | $-1$ | $\begin{aligned} & D_{y_{10}}+D_{x_{01}}+X \rightarrow x_{10} \\ & -(A y)_{10}+-(A x)_{10}+X \rightarrow-(A x)_{10} \end{aligned}$ | Add BCD source and eXtend bit to destination，BCD result |
| ADD ${ }^{4}$ | BWL | s，Dn <br> Dn，d | ＊＊＊＊＊ | $\begin{aligned} & \mathrm{e} \\ & \mathrm{~B} \end{aligned}$ | $\begin{array}{\|c} \mathrm{s} \\ \mathrm{~d}^{4} \end{array}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | s | $s$ | $s^{4}$ | $\begin{aligned} & s+D_{n} \rightarrow D_{n} \\ & D_{n}+d \rightarrow d \end{aligned}$ | Add binary（ADDI or ADDD is used when source is \＃n．Prevent ADDI with \＃n．l） |
| ADDA ${ }^{4}$ | WL | s，An |  | 5 | E | 8 | $\delta$ | s | $s$ | 8 | $s$ | $s$ | $s$ | $s$ | s | $s+A n \rightarrow A n$ | Add address（．W sign－extended to．L） |
| ADDI ${ }^{4}$ | BWL | \＃n，d | ＊＊＊＊＊ | d | － | d | d | d | d | d | d | d | － | － | s | $\# \mathrm{n}+\mathrm{d} \rightarrow \mathrm{d}$ | Add immediate to destination |
| $\mathrm{ADOL}^{4}$ | BWL | \＃n，d | ＊＊＊＊＊ | d | d | d | d | d | d | d | d | d | － | － | S | $\# \mathrm{n}+\mathrm{d} \rightarrow \mathrm{d}$ | Add quick immediate（\＃п range：I to 8） |
| ADDX | BWL | $\begin{aligned} & \text { Dy, Dx } \\ & -(A y)-(A x) \end{aligned}$ | ＊＊＊＊＊ | E |  |  | － | 8 | － | － | － | － | － | － |  | $\begin{aligned} & D y+D x+X \rightarrow D x \\ & -(A y)+-(A x)+X \rightarrow-(A x) \end{aligned}$ | Add source and eXtend bit to destination |
| AND ${ }^{4}$ | BWL | s．Dn <br> Dn，d | －＊＊00 | $\begin{aligned} & \mathrm{e} \\ & \mathrm{e} \end{aligned}$ |  | $\begin{array}{r} \mathrm{s} \\ \mathrm{~d} \\ \hline \end{array}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | s | $s$ | $s^{4}$ | $\begin{aligned} & \mathrm{s} \text { AND Dn } \rightarrow \mathrm{Dn}_{\mathrm{n}} \\ & \mathrm{Dn}_{\mathrm{n}} \text { AND d } \rightarrow \mathrm{d} \end{aligned}$ | Logical AND source to destination （ANDI is used when source is \＃n） |
| ANOI $^{4}$ | BWL | \＃n，d | －＊＊00 | d | － | d | d | d | d | d | d | d | － | － | s | $\#$ AND d $\rightarrow$ d | Logical AND immediate to destination |
| $\mathrm{ANOI}^{4}$ | $B$ | \＃n，LCR | 三트프틀 | － | － | － | － | － | － | － | － | － | － | － | $\delta$ | \＃n AND CLR $\rightarrow$ CLR | Logical AND immediate to CLR |
| ANDI $^{4}$ | W | \＃n，SR | 크트틍 | － | － | － | － | － | － | － | － | － | － | － | s | \＃n AND SR $\rightarrow$ SR | Logical AND immediate to SR（Privileged） |
| $\begin{aligned} & \text { ASL } \\ & \text { ASR } \end{aligned}$ | $\begin{gathered} \hline \text { BWL } \\ \text { W } \end{gathered}$ | $\begin{aligned} & D x, D y \\ & \# n, D y \\ & \text { d } \end{aligned}$ | ＊＊ | $\begin{aligned} & \mathrm{e} \\ & \mathrm{~d} \end{aligned}$ |  |  | d |  |  | $\mathrm{d}$ |  | d |  |  | s | $\rightarrow$ | Arithmetic shift Dy by Dx bits left／right Arithmetic shift Dy \＃n bits L／R（\＃n：Ito 8） Arithmetic shift ds I bit left／right（．W only） |
| Bcc | BW ${ }^{3}$ | address ${ }^{2}$ |  | － | － | － | － | － | － | － | － | － | － | － | － | if ec true then address $\rightarrow$ Р | Branch conditionally（ce table on back） （8 or lf－bit $\pm$ offset to address） |
| BLHG | B L | Dn，d <br> \＃n，d | －－ | $\begin{aligned} & e^{1} \\ & d^{1} \end{aligned}$ |  | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ |  |  | s | NOT（bit number of d）$\rightarrow$ Z NDT（bit n of d）$\rightarrow$ bit $n$ of $d$ | Set $Z$ with state of specified bit in d then invert the bit in d |
| BCLR | B L | Dn，d \＃n，d | －－＊－－ | $\begin{aligned} & \mathrm{e}^{\mathrm{I}} \\ & \mathrm{~d}^{\mathrm{d}} \end{aligned}$ |  | $\begin{aligned} & \text { d } \\ & d \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ |  |  | s | $\begin{aligned} & \text { NDT(bit number of } d \text { ) } \rightarrow z \\ & \square \rightarrow \text { bit number of } d \end{aligned}$ | Set $Z$ with state of specified bit in d then clear the bit ind |
| BRA | BW ${ }^{3}$ | address ${ }^{2}$ | －－－－－－ | － | － | － | － | － | － | － | － | － | － | － | － | address $\rightarrow$ Р［ | Branch always（8 or li－bit $\pm$ offset to addr） |
| BSET | B L | Dn，d \＃n，d | －－＊ | $\begin{aligned} & \mathrm{e} \\ & \mathrm{~d}^{\prime} \end{aligned}$ |  | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ |  |  | $s$ | $\begin{aligned} & \text { NOT( bit } n \text { of } d) \rightarrow z \\ & 1 \rightarrow \text { bit } n \text { ofd } \end{aligned}$ | Set $Z$ with state of specified bit in d then set the bit in d |
| BSR | $B W^{3}$ | address ${ }^{2}$ | －－－－－－ | － | － | － | － | － | － | － | － | － | － | － | － | PC $\rightarrow$－（SP）；address $\rightarrow$ P［ | Branch to subroutine（8 or 16 －bit $\pm$ offset） |
| BIST | B L | Dn，d <br> \＃n，d | －－ | $\begin{aligned} & \mathrm{e}^{\mathrm{I}} \\ & \mathrm{~d}^{\prime} \end{aligned}$ |  | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~d} \end{aligned}$ | $\mathrm{s}$ | $\begin{aligned} & \text { NDT( bit Dn of } d) \rightarrow z \\ & \text { NOT(bit \#n of } d) \rightarrow z \end{aligned}$ | Set $Z$ with state of specified bit in d Leave the bit ind unchanged |
| CHK | W | s，Dn | －＊UUU | E | － | s | s | s | s | s | s | s | s | $s$ | s | if $\mathrm{Dn}_{\sim}<\square_{\text {ar }} \mathrm{D}_{n}>$ s then TRAP | Compare Dn with प and upper bound［s］ |
| CLR | BWL | d | －0100 | d | － | d | d | d | d | d | d | d | － | － | － | $\square \rightarrow$ d | Clear destination to zero |
| CMP ${ }^{4}$ | BWL | s，Dn | －＊＊＊＊ | E | $\mathrm{s}^{4}$ | s | $s$ | $s$ | s | s | s | s | s | $s$ | $\mathrm{s}^{4}$ | set CLR with $\mathrm{D}_{\mathrm{n}}-\mathrm{s}$ | Compare Dn to saurce |
| CMPA $^{4}$ | WL | s．An | －＊＊＊＊ | s | E | 8 | $s$ | s | 5 | $s$ | s | s | $s$ | $s$ | s | set CLR with An－s | Compare An to source |
| CMP1 ${ }^{4}$ | BWL | \＃n，d | －＊＊＊＊ | d | － | d | d | d | d | d | d | d | － | － | s | set CLR withd－\＃n | Compare destination to \＃n |
| CMPM $^{4}$ | BWL | （Ay）＋，（Ax）＋ | －＊＊＊＊ | － | － | － | e | － | － | － | － | － | － | － | － | set CLR with（Ax）－（Ay） | Compare（Ax）to（Ay）；Increment Ax and Ay |
| DBcc | W | Dn，addres ${ }^{2}$ | －－－－－－ | － | － | － | － | － | － | － | － | － | － | － | － | if ce false then $\left\{\mathrm{Dn}_{\mathrm{n}} \mathrm{I} \rightarrow \mathrm{D}_{\mathrm{n}}\right.$ <br> if $\mathrm{Dn}_{\mathrm{n}}$ く＞－ 1 then addr $\rightarrow$ P $\}$ | Test condition，decrement and branch （1F－bit $\pm$ offset to address） |
| DIVS | W | s，Dn | －＊＊＊0 | E | － | s | s | s | s | s | s | s | s | $s$ | 5 | $\pm 32 \mathrm{bit} \mathrm{Dn} / \pm$｜6bit s $\rightarrow \pm \mathrm{Dn}^{\text {n }}$ | Dn n ［［16－bit remainder，l6－bit quatient ］ |
| DIVI | W | s，Dn | －＊＊＊0 | E | － | s | 8 | $s$ | $s$ | 8 | $s$ | s | s | 8 | s | 32 bit Dn／估its $\rightarrow$ Dn | $\mathrm{D}_{\mathrm{n}}=$［ 16－bit remainder，If－bit quotient ］ |
| ERR ${ }^{4}$ | BWL | Dn，d | －＊＊00 | E | － | d | d | d | d | d | d | d | － | － | $\mathrm{s}^{4}$ | Dn X $\mathrm{DR} \mathrm{d} \rightarrow$ d | Logical exclusive DR Dn to destination |
| EDR1 ${ }^{4}$ | BWL | \＃n，d | －＊＊00 | d | － | d | d | d | d | d | d | d | － | － | s | \＃n XDR d $\rightarrow$ d | Logical exclusive IR \＃n to destination |
| EDR ${ }^{4}$ | B | \＃n．LCR |  | － | － | － | － | － | － | － | － | － | － | － | s | \＃n XDR RCR $\rightarrow$ CLR | Logical exclusive DR \＃n to CCR |
| EDRI ${ }^{4}$ | W | \＃n，SR | \＃\＃\＃\＃\＃ | － | － | － | － | － | － | － | － | － | － | － | $s$ | \＃n XDR SR $\rightarrow$ SR | Logical exclusive IR \＃n to SR（Privileged） |
| EXI | L | Rx，Ry | －－－－－ | 8 | E | － | － | － | － | － | － | － | － | － | － | register $\leftarrow \rightarrow$ register | Exchange registers（32－bit only） |
| EXT | WL | Dn | －＊＊00 | d | － | － | － | － | － | － | － | － | － | － | － |  | Sign extend（change ．B to．W or．W to．L） |
| ILLEGAL |  |  | －－－－－ | － | － | － | － | － | － | － | － | － | － | － | － | P $\rightarrow$－（SSP）；SR $\rightarrow$－（SSP） | Generate Illegal Instruction exception |
| JMP |  | d |  | － | － | d | － | － | d | d | d | d | d | d | － | $\uparrow d \rightarrow$ 「L | Jump to effective address of destination |
| JSR |  | d | －－－－－ | － | － | d | － | － | d | d | d | d | d | d | － | 䦻 $\rightarrow$－（SP）：$\uparrow$ d $\rightarrow$ 䦻 | push Pए，jump to subroutine at address d |
| LEA | L | s．An | －－－－－ | － | E | S | － | － | 8 | 8 | 8 | s | s | $s$ | － | $\uparrow_{\mathrm{s}} \rightarrow \mathrm{An}$ | Load effective address of $s$ to An |
| LINK |  | An，\＃n | －－－－－ | － | － | － | － | － | － | － | － | － | － | － | － | $\begin{aligned} & A n \rightarrow-(S P) ; S P \rightarrow A n ; \\ & S P+\# n \rightarrow S P \end{aligned}$ | Create local warkspace on stack （negative $n$ to allocate space） |
| $\begin{aligned} & \text { LSL } \\ & \text { LSR } \end{aligned}$ | BWL | $\begin{aligned} & D x, D y \\ & \# n, D y \\ & \text { d } \end{aligned}$ | ＊＊＊0＊ | $\begin{aligned} & \mathrm{e} \\ & \mathrm{~d} \end{aligned}$ | － |  |  |  |  | $\mathrm{d}$ |  |  |  | - | s | $\underset{0 \rightarrow 4}{x} \longrightarrow$ | Logical shift Dy．Dx bits left／right Logical shift Dy．\＃n bits L／R（\＃n：I to B） Lugical shift d I bit left／right（．W only） |
| MIVE $^{4}$ | BWL | s．d | －＊＊00 | E | $s^{4}$ | 8 | 8 | E | E | E | 8 | E | s | 8 | $\mathrm{s}^{4}$ | $s \rightarrow$ d | Move data from source to destination |
| MDVE | W | s．CLR | 트ㅌㅡㅡㅔ | $\Sigma$ | － | 8 | 8 | s | s | S | 5 | $\delta$ | $s$ | 5 | 8 | $s \rightarrow$ CLR | Move source to Condition Code Register |
| MDVE | W | s，SR |  | 5 | － | s | $s$ | 5 | 5 | 8 | 5 | s | s | $s$ | s | $s \rightarrow$ SR | Move source to Status Register（Privileged） |
| MIVE | W | SR，d | －－－－－ | d | － | d | d | d | d | d | d | d | － | － | － | SR $\rightarrow$ d | Move Status Register to destination |
| MDVE | L | $\begin{aligned} & \text { USP.An } \\ & \text { An,USP } \end{aligned}$ | －－－－－ |  | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~s} \end{aligned}$ |  | － | － | － | － | － | － | － | － |  | $\begin{aligned} & U S P \rightarrow A n \\ & A n \rightarrow U S P \end{aligned}$ | Move User Stack Pointer to An（Privileged） Move An to User Stack Pointer（Privileged） |
|  | BWL | s．d | XNZVC | Dn | An | （An） | （An）＋ | －（An） | （i．An） | （i．An．Rn） | abs．W | abs．L | （i．PC） | （i， $\mathrm{F} \mathrm{C}, \mathrm{Rn}$ ） | \＃n |  |  |

Computer Architecture - EPITA - S3 - 2020/2021

| Upcade | Sizs | Uperana | Liek |  | tttec | tive | adares | $s \mathrm{~s}=\mathrm{s}$ | ource, | =dest | tion, | sither | i=0 | lacement |  | Uperation | Uescriptian |
| :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  | BWL | s.d | XNZVC | Dn | An | (An) | (An)+ | -(An) | (i.An) | (i,An.Rn) | abs.W | abs.L | (i.PC) | (i.PL.Rn) | \#n |  |  |
| MDVEA ${ }^{4}$ | WL | s,An | ----- | $\Sigma$ | 8 | s | s | s | s | $s$ | $s$ | s | $s$ | $s$ | s | $s \rightarrow A n$ | Move source to An (MDVE s,An use MDVEA) |
| MIVEM ${ }^{4}$ | WL | Rn-Rn,d <br> s.Rn-Rn |  |  |  | $\mathrm{d}$ | $\mathrm{s}$ | $\bar{d}$ | $\mathrm{d}$ | $\mathrm{d}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~s} \end{aligned}$ | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~s} \end{aligned}$ | $\Sigma$ | $s$ |  | $\begin{aligned} & \text { Registers } \rightarrow \mathrm{d} \\ & s \rightarrow \text { Registers } \end{aligned}$ | Move specified registers to/from memory (.W source is sign-extended to . L for Rn ) |
| MDVEP | WL | Dn,(i,An) (i.An).Dn |  | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | - |  |  | - | $\begin{aligned} & \mathrm{d} \\ & \mathrm{~s} \end{aligned}$ | - | - |  | - |  |  | $\begin{aligned} & D_{n \rightarrow} \rightarrow(\text { (i,An }) \ldots(i+2, A n) \ldots(i+4, A . \\ & (i, A n) \rightarrow D_{n} . .(i+2, A n) \ldots(i+4, A . \end{aligned}$ | Move $\mathrm{Dn}_{\mathrm{n}}$ /from alternate memory bytes (Access only even ar add addresses) |
| MDVE® ${ }^{4}$ | L | \#n, Dn | -**00 | $d$ | - | - | - | - | - | - | - | - | - | - | $\Sigma$ | $\# \mathrm{n} \rightarrow \mathrm{D}_{\mathrm{n}}$ | Mave sign extended 8-bit \#n to Dn |
| MULS | W | s, Dn | -**00 | B | - | s | s | s | S | $s$ | s | s | s | s | $\Sigma$ | $\pm$ IGbit s ${ }^{*} \pm$ IShit $\mathrm{Dn}_{n} \rightarrow \pm \mathrm{Dn}^{\text {n }}$ | Multiply signed If-bit; result: signed 32-bit |
| MULU | W | s, Dn | -**00 | E | - | s | s | s | S | s | s | s | S | $s$ | s |  | Multiply unsig'd I6-bit; result: unsig'd 32-bit |
| NBCD | B | d | *U*U* | d | - | d | d | d | d | d | d | d | - | - | - | $\square-\mathrm{d}_{0}-\mathrm{X} \rightarrow \mathrm{d}$ | Negate BCD with eXtend. BCD result |
| NEE | BWL | d | ***** | d | - | d | d | d | d | d | d | d | - | - | - | $\square-\mathrm{d} \rightarrow \mathrm{d}$ | Negate destination (2's complement) |
| NEEX | BWL | d | *** | d | - | d | d | d | d | d | d | d | - | - | - | $\square-\mathrm{d}-\mathrm{X} \rightarrow \mathrm{d}$ | Negate destination with eXtend |
| NIP |  |  | ----- | - | - | - | - | - | - | - | - | - | - | - | - | None | No operation occurs |
| NDT | BWL | d | -**00 | d | - | d | d | d | d | d | d | d | - | - | - | $\mathrm{NDT}(\mathrm{d}) \rightarrow \mathrm{d}$ | Logical NDT destination (I's complement) |
| DR ${ }^{4}$ | BWL | $\begin{aligned} & \mathrm{s}, \mathrm{Dn}_{\mathrm{n}} \\ & \mathrm{Dn}_{\mathrm{n}, \mathrm{~d}} \end{aligned}$ | -**00 | $\begin{aligned} & \text { E } \\ & \text { e } \end{aligned}$ | - | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \text { d } \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \text { s } \\ & \text { d } \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | s | s | $s^{4}$ | $\begin{aligned} & \mathrm{s} \text { 听 } \mathrm{Dn}_{\mathrm{n}} \rightarrow \mathrm{Dn}_{\mathrm{n}} \\ & \mathrm{Dn}_{\mathrm{n}}^{\mathrm{DR} \mathrm{~d} \rightarrow \mathrm{~d}} \\ & \hline \end{aligned}$ | Logical IR <br> (DRI is used when source is \#n) |
| DR14 | BWL | \#n,d | -**00 | d | - | d | d | d | d | d | d | d | - | - | $s$ | \#n ПRd $\rightarrow$ d | Lugical DR \#n to destination |
| DRI ${ }^{4}$ | B | \#n.CLR |  | - | - | - | - | - | - | - | - | - | - | - | $s$ | \#n IR CLR $\rightarrow$ CLR | Logical DR \#n to CLR |
| DRI ${ }^{4}$ | W | \#n,SR | \#\#\#\#\# | - | - | - | - | - | - | - | - | - | - | - | s | \#n DR SR $\rightarrow$ SR | Logical IR \#n to SR (Privileged) |
| PEA | [ | s | ----- | - | - | s | - | - | s | 8 | $\delta$ | 8 | $s$ | $s$ | - | $\mathrm{T}_{\mathrm{s} ~}^{\text {}}$-(SP) | Push effective address of s onto stack |
| RESET |  |  | ----- | - | - | - | - | - | - | - | - | - | - | - | - | Assert RESET Line | Issue a hardware RESET (Privileged) |
| $\begin{array}{\|l\|} \hline R D L \\ R D R \\ \hline \end{array}$ | $\begin{gathered} \hline \text { BWL } \\ \text { W } \end{gathered}$ | $\begin{aligned} & \mathrm{Dx}, \mathrm{Dy} \\ & \# n, D y \\ & \mathrm{~d} \end{aligned}$ | -**0* | $\begin{aligned} & \text { e } \\ & \text { d } \end{aligned}$ |  |  | d | d | d | $\mathrm{d}$ |  |  |  | - | s | $\stackrel{\square}{\square} \square$ | Rotate Dy. Dx bits left/right (without X) Rotate Dy, \#n bits left/right (\#n: I to 8) Rotate dI-bit left/right (.W only) |
| RIXL <br> RIXR | BWL <br> W | $\begin{aligned} & \hline x_{x, D y} \\ & \# n, D y \\ & \text { d } \end{aligned}$ | ***0* | $\begin{aligned} & \text { e } \\ & \text { d } \end{aligned}$ |  |  | $d$ | $\mathrm{d}$ | $\mathrm{d}$ | d |  |  |  |  | $s$ | $\xrightarrow{c \rightarrow c}$ | Rotate Dy, Dx bits L/R, Xused then updated Rotate Dy. \#n bits left/right (\#n: I to B) <br> Rotate destination I-bit left/right (.W only) |
| RTE |  |  |  | - | - | - | - | - | - | - | - | - | - | - | - | (SP) $+\rightarrow$ SR; (SP) $+\rightarrow$ P | Return from exception (Privileged) |
| RTR |  |  | 프=ㅌㅡㅡㄹ | - | - | - | - | - | - | - | - | - | - | - | - | (SP) $+\rightarrow$ CLR. (SP) $+\rightarrow$ P[ | Return from subroutine and restore CLR |
| RTS |  |  | ----- | - | - | - | - | - | - | - | - | - | - | - | - | (SP) $+\rightarrow$ P[ | Return from subrautine |
| SBCD | B | $\begin{aligned} & \text { Dy.Dx } \\ & -(A y)-(A x) \end{aligned}$ | * ${ }^{*} \mathrm{U}^{*}$ | e |  |  |  | E |  | - | - |  |  |  |  | $\begin{aligned} & D x_{10}-D y_{10}-x \rightarrow D x_{10} \\ & -(A x)_{10}-(A y)_{10}-X \rightarrow-(A x)_{10} \end{aligned}$ | Subtract BCD source and eXtend bit from destination, BCD result |
| Sce | B | d |  | d | - | d | d | d | d | d | d | d | - | - | - | $\begin{array}{r} \text { If ec is true then I's } \rightarrow \mathrm{d} \\ \text { else I's } \rightarrow \mathrm{d} \end{array}$ | $\begin{aligned} \text { If ce true then } \mathrm{d} . \mathrm{B} & =11111111 \\ \text { else } \mathrm{d} . \mathrm{B} & =00000000 \end{aligned}$ |
| STIP |  | \#n | ==ㅡ= | - | - | - | - | - | - | - | - | - | - | - | 5 | \#n $\rightarrow$ SR; STIP | Mave \#n to SR, stap processor (Privileged) |
| SUB ${ }^{4}$ | BWL | $\begin{array}{\|l} \hline \text { s, Dn } \\ D_{n, d} \end{array}$ | ***** | $\begin{aligned} & \mathrm{E} \\ & \mathrm{E} \end{aligned}$ | $\begin{gathered} 5 \\ d^{4} \\ \hline \end{gathered}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \end{aligned}$ | $\begin{aligned} & \mathrm{s} \\ & \mathrm{~d} \\ & \hline \end{aligned}$ | s | $\mathrm{s}$ | $\mathrm{s}^{4}$ | $\begin{aligned} & D_{n}-\mathrm{s} \rightarrow \mathrm{D}_{\mathrm{n}} \\ & \mathrm{~d}-\mathrm{Dn}_{\mathrm{n}} \rightarrow \mathrm{~d} \end{aligned}$ | Subtract binary (SUBI ar SUBR used when source is \#n. Prevent SUBQ with \#n.l) |
| SUBA ${ }^{4}$ | WL | s,An | ------ | s | 8 | s | 8 | S | s | S | $s$ | s | s | s | $\delta$ | An $-s \rightarrow$ An | Subtract address (.W sign-extended to .L) |
| SUBI ${ }^{4}$ | BWL | \#n,d | ***** | d | - | d | d | d | d | d | d | d | - | - | 8 | $\mathrm{d}-\# \mathrm{n} \rightarrow \mathrm{d}$ | Subtract immediate from destination |
| SUBE ${ }^{\text {a }}$ | BWL | \#n,d |  | d | d | d | d | d | d | d | d | d | - | - | 8 | $\mathrm{d}-\# \mathrm{n} \rightarrow \mathrm{d}$ | Subtract quick immediate (\#n range: I to 8) |
| SUBXX | BWL | $\begin{aligned} & \text { Dy.Dx } \\ & -(A y) .-(A x) \end{aligned}$ | ***** | e |  |  |  | E |  | - | - |  |  |  |  | $\begin{aligned} & D x-D y-X \rightarrow D x \\ & -(A x)--(A y)-X \rightarrow-(A x) \end{aligned}$ | Subtract source and eXtend bit from destination |
| SWAP | W | Dn | -**00 | d | - | - | - | - | - | - | - | - | - | - | - | bits[3I:LB] $\longrightarrow \rightarrow$ bits[ $15: D]$ | Exchange the lf-bit halves of Dn |
| TAS | B | d | -**00 | d | - | d | d | d | d | d | d | d | - | - | - | test d $\rightarrow$ CLR; $1 \rightarrow$ bit7 of $d$ | N and Z set to reflect d , bit7 of d set tol |
| TRAP |  | \#n | ----- | - | - | - | - | - | - | - | - | - | - | - | $\delta$ | $\begin{aligned} & \text { P丁-(SSP);SR } \rightarrow \text {-(SSP); } \\ & \text { (vector table entry) } \rightarrow \text { P } \end{aligned}$ | Push PC and SR, PC set by vector table \#n (\#n range: D to 15) |
| TRAPV |  |  | -- | - | - | - | - | - | - | - | - | - | - | - | - | If V then TRAP \#7 | If averflow, execute an Dverflow TRAP |
| IST | BWL | d | -**00 | d | - | d | d | d | d | d | d | d | - | - | - | test d $\rightarrow$ CLR | $N$ and Zset to reflect destination |
| UNLK |  | An | ------ | - | d | - | - | - | - | - | - | - | - | - | - | $\mathrm{An}^{\rightarrow} \mathrm{SP} ;(\mathrm{SP})+\rightarrow \mathrm{An}$ | Remove local workspace from stack |
|  | BWL | s.d | XNZVC | Dn | An | (An) | (An)+ | -(An) | (i,An) | (i,An,Rn) | abs.W | abs.L | (i.PC) | (i.PC.Rn) | \#n |  |  |


| Condition Tests ( + IR, INOT, $\oplus$ XIR:, "Unsigned, 'Alternate cr ) |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| ${ }_{\text {ct }}$ | Condtition | Test | cr | Condition | Test |
| T | true | 1 | , | overflow lear | IV |
| F | false | 0 | Vs | overflow set | V |
| ${ }^{\text {H }}$ | higher than | $1(C+2)$ | Pl | plus | IN |
| $15^{\circ}$ | lower or same | [+1 | M1 | minus | N |
| HS ${ }^{4}$. $\mathrm{CL}^{\text {a }}$ | higher or same | 10 | ${ }^{6}$ | greater or equal | $!(\mathbb{\oplus} \oplus$ V) |
| LTa $\mathrm{CS}^{\text {c }}$ | lower than | ᄃ | IT | less than | $(\mathrm{N} \oplus \mathrm{V})$ |
| NE | not equal | 12 | GT | greater than | $\underline{[ }(\mathrm{N} \oplus \mathrm{V})+2]$ |
| Ea | equal | 2 | LE | less or equal | $(\mathrm{N} \oplus \mathrm{V})+\mathrm{l}$ |

Revised by Peter Csaszar, Lawrence Tech University - 2004-2006

An Address register ( $(16 / 32$-bit, $n=\square-7$ )
In Data register ( $8 / \mathrm{I} / \mathrm{B} / 32-$ bit, $n=\square-7$ )
Rn any data or address register
s Source, d Destination
e Either source or destination
\#n Immediate data, I Displacement
BCD Binary Coded Decimal
$\uparrow$ Effective address
Long only: all others are byte only Assembler calculates affset
Branch sizes: $\mathbf{B}$ or $. S-128$ to +127 bytes, .W or $. \mathrm{L}-32788 \mathrm{to}+32767$ bytes
Assembler autamatically uses A. I. © ar M form if possible. Use \#n.L to prevent Quick optimization

[^0]Last name:
First name:
Group:

## ANSWER SHEET TO BE HANDED IN

## Exercise 1

| Instruction | Memory | Register |
| :---: | :---: | :---: |
| Example | \$005000 54 AF 0040 E7 2148 C0 | $\begin{aligned} & \mathrm{A} 0=\$ 00005004 \\ & \mathrm{~A} 1=\$ 0000500 \mathrm{C} \end{aligned}$ |
| Example | \$005008 C9 1011 C8 D4 36 FF 88 | No change |
| MOVE.W \#5000, -4(A2) |  |  |
| MOVE.L \$500A, - (A2) |  |  |
| MOVE.B 1(A0),-7(A1, D1.W) |  |  |
| MOVE.L - 12(A2), \$13(A1, D2.W) |  |  |

## Exercise 2

| Operation | Size <br> (bits) | Result <br> (hexadecimal) | $\mathbf{N}$ | $\mathbf{Z}$ | $\mathbf{V}$ | $\mathbf{C}$ |
| :--- | :---: | :---: | :---: | :---: | :---: | :---: |
| $\$ 42+\$ 2 \mathrm{~A}$ | 8 |  |  |  |  |  |
| $\$ 7 F F 0+\$ 11$ | 16 |  |  |  |  |  |
| $\$ F F F F F F F 0+\$ 11$ | 32 |  |  |  |  |  |

## Exercise 3

| Values of registers after the execution of the program. <br> Use the 32-bit hexadecimal representation. |  |
| :---: | :---: |
| D1 = \$ | D3 $=\$$ |
| D2 = \$ | D4 $=\$$ |

ToAscii

ToInt

[^1]
[^0]:    Distributed under the GNU general public use license.

[^1]:    GetSum

