|  |  |
| --- | --- |
| **Aufgabe** | **Befehl** |
| Anzeige aller Dateinamen der Dateien im aktuellen Arbeitsverzeichnis | ls |
| Anzeige aller Dateinamen der Dateien im aktuellen Arbeitsverzeichnis | ls -l |
| Erstellung einer Datei „heute“ mit dem aktuellen Datum und der aktuellen Uhrzeit  durch einen Shell-Befehl (→ Ausgabeumlenkung!) | date > heute |
| Erstellung einer Datei „prozessbaum“ mit einer Baumdarstellung aller aktuellen Prozesse  inkl. Ausgabe der Prozess-IDs (→ Ausgabeumlenkung!) | ps > prozessbaum |
| Anzeige der Datei „prozessbaum“ mit seitenweiser Ausgabe | more prozessbaum |
| Anzeige aller Dateinamen der Dateien im aktuellen Arbeitsverzeichnis,  wobei nur Dateien, die mit dem Buchstaben „p“ beginnen, angezeigt werden. | find ./ -name „p\*“ |
| Erstellen eines neuen Verzeichnisses „meindir“ | mkdir meindir |
| Verschieben der Datei „prozessbaum“ in das Unterverzeichnis „meindir | mv prozessbaum meindir |
| Suchen und Ausgeben aller Dateipfade von Dateien, die mit „p“ beginnen und sich im aktuellen  Verzeichnis oder einem Unterverzeichnis befinden | find ./ -name „p\*“ -print |
| Wechseln des Arbeitsverzeichnisses zu „meindir“ Kopieren der Datei „prozessbaum“ in eine  neue Datei mit dem Namen „Neuer Prozessbaum“ im selben Verzeichnis | cd /meindir/ | cp prozessbaum „Neuer Prozessbaum“ |

**Aufgabe 6**

# **Aufgabe 4 (Von-Neumann-Maschine)**

## **Gerade / Ungerade**

## **Potenz berechnen**

|  |  |
| --- | --- |
| **ld 1 ;create counter**  **st $66**  **in $64 ;enter base**  **ld $64**  **st $67**  **in $65 ;enter potenz**  **;check Potenz > 0**  **ld $65**  **cmp 0**  **jgt LoopStart**  **ld 1**  **st $67**  **jmp ende**  **;Loop**  **;compare Counter**  **ld $66 :loopStart**  **cmp $65**  **jeq ende**  **add 1**  **st $66**  **;execute multiply**  **ld $67**  **mul $64**  **st $67**  **jmp loopStart**  **;Output and end**  **out $67 :ende**  **end** | **Erstellt einen Counter, um später mit der Potenz zu vergleichen.** |
| **Speichert Counter in den Speicherplatz 66** |
| **Eingabe des Basiswerts und Sicherung des Werts in Adresse 64** |
| **Lädt Basiswert und speichert den Basiswert in Adresse 64** |
| **Erstellt eine Kopie des Basiswerts in der Adresse 67** |
| **Eingabe des Potenzwertes und Sicherung in 65** |
|  |
| **//Abschnitt Überprüfung der Potenz.**  **Lädt den Potenzwert aus Adresse 65.**  **Vergleicht den Wert mit 0**  **Ist der Werte größer als 0, wird zum Marker LoopStart gesprungen**  **Ist der Wert = 0, dann wird die Zahl 1 geladen**  **Die Zahl 1 wird in den Speicher mit Adresse 67 geschrieben.**  **Nach Ausführung wird an das Ende der Befehlsgruppe gesprungen.**  **//Abschnitt: Loop – Bedingungsüberprüfung.**  **Lädt den Counter aus Adresse 66**  **Vergleicht den geladenen Wert mit der Potenz in Adresse 65**  **Hat der Counter den gleichen Wert wie die Potenz, Sprung ans Ende.**  **Ist der Counter kleiner als die Potenz, wird zum Counter 1 addiert.**  **Der neue Wert des Counters wird in die Adresse 66 gespeichert.**  **//Abschnitt: Loop – Ausführung der Potenzrechnung**  **Die Kopie des Basiswerts wird aus Adresse 67 geladen.**  **Die Kopie im Akku wird nun mit der Basis aus Adresse 64 multipliziert.**  **Das Ergebnis der Operation überschreibt nun die Kopie -> Ergebnis.**  **Sprung zurück an den Anfang des Loops**  **// Abschnitt: Ausgabe Schleifen wurden verlassen. Der Wert in Adresse 67 wird ausgegeben.**  **Ende.** |