GSS Blatt 2

Aufgabe 1

a)

Die eine Grundaufgabe eines Betriebssystems ist die Erweiterung der Hardware, wobei es darum geht, alle **Hardwarekomponenten** (wie z.B. Prozessor, Festplatte, RAM, Grafikkarte) insofern **zu erweitern**, als dass auf den oberen Ebenen einfacher programmiert werden kann und nicht alle Eventualitäten vom Hochsprachenprogrammierer mühsam abgedeckt und berücksichtigt werden müssen. Der Programmierer wünscht sich somit eine Abstraktion der unteren Ebenen.

Die andere Grundaufgabe eines Betriebssytems besteht darin die verfügbaren **Ressourcen** sinnvoll **zu managen**. Das Betriebssystem sorgt für einen geordneten Ereignisablauf, sodass beispielsweise Prozessoren, Speicher sowie Ein- und Ausgabegeräte sich gegenseitig nicht im Weg stehen und kein Chaos entsteht. Auch laufende Prozesse wollen priorisiert werden, sodass wichtige Ereignisse vorrangig zu weniger wichtigen Ereignisse geschehen.

b)

Erweiterung der Hardwarekomponenten sorgt für

- Verwaltung von Betriebsmitteln
- Erhöhung der Benutzbarkeit
- Erleichterung menschlicher Kommunikation

Ressourcenmanagment sorgt für

- Minderung der Maschinenabhängigkeit
- Erhöhung der Zuverlässigkeit
- automatische Ausführung
- Buchhaltung

Aufgabe 2

a)

Programm

Ein Programm ist ein Resultat eines kompilierten Quellcodes, das geschaffen wurde, um ein bestimmtes Problem zu lösen oder eine Aufgabe wahrzunehmen.

Prozess

Ein Prozess ist eine Abstraktion eines laufenden Programms. Wenn es um Prozesse geht, spricht man auch von “Quasiparallelität”, da der Benutzer beispielsweise drei Prozesse sieht, die scheinbar parallel, in Wahrheit aber sequentiell bearbeitet werden. Moderne Betriebssysteme sind dermaßen gestaltet, dass es für Menschen einfacher ist, über Parallelität den Überblick zu behalten. Multiprozessorsysteme, die echte Parallelität mit sich bringen, sind ungebräuchlich und werden von den meisten Anwendern tendenziell nicht in Erwägung gezogen.

Thread

Threads sind sozusagen Subprozesse von Prozessen, die es ermöglichen innerhalb einer geschriebenen Software Parallelität (bzw. Quasiparallelität) zu erreichen. Mit Hilfe von ihnen kann beispielsweise aus einem einzigen Prozess ein Softwaregebilde entstehen, in dem spezielle Aufgaben auf Threads ausgelagert werden können und somit eine ggf. notwendige Effizienz geschaffen wird. Aus Programmierersicht sind Threads sehr nützlich, wünschenswert und vielseitig einsetzbar.

d)

X: Ready/Waiting – Ein wartender Prozess liegt im Hauptspeicher und erwartet, ausgeführt zu werden. Es ist möglich, mehrere wartende Prozesses zu haben.

Y: Running – Ein Prozess im rechnenden Zustand wird momentan vom System ausgeführt/berechnet. Pro Kern kann nur ein Prozess ausgeführt werden.

Z: Blocked – Ein Prozess blockiert, falls er momentan nicht mehr ausgeführt werden kann, aber noch nicht vollständig bearbeitet (und damit auch nicht bereit zum Terminieren) ist.

a: Admit/Entry – Ein neuer Prozess wird vom System erkannt und in den Hauptspeicher geladen.

b: Dispatch – Ein wartender Prozess wird zum rechnenden Zustand versendet, sobald er der erste Prozess in der Warteschlange eines Kernes ist.

c: Event wait – Falls ein momentan rechnender Zustand auf ein Ereignis warten muss, bevor er weiter bearbeitet werden kann, wird er in den blockierten Zustand bewegt.

d: Event completion – Sobald das Ereignis, das von einem blockierten Prozess erwartet wird, eintritt, wird dieser Prozess wieder in den wartenden Zustand versetzt.

e: Release – Sobald ein Prozess im rechnenden Zustand vollständig bearbeitet wurde, wird er vom System terminiert.

f: Timeout/Yield - Ein rechnender Prozess, der seine vom System zugeteilte Bearbeitungszeit überschreitet, wird wieder in den wartenden Zustand bewegt.

Aufgabe 3

a)

Annahme: Überdeckung des 2. Operanden; s1 und s2 sind zusätzliche Speicherzellen; R ist das Zielregister für das Endergebnis

BEFEHL		WIRKUNG
MOVE	a1, s1	s1 := a1
ADD	a2, s1	s1 := a1 + a2
DIV	a3, s1	s1 := (a1 + a2) / a3
MOVE	b1, s2	s2 := b1
SUB	b2, s2	s2 := b1 – b2
DIV	b3, s2	s2 := (b1 – b2) / b3
ADD	s2, s1	s1 := ((a1 + a2) / a3) + ((b1 – b2) / b3)
MOVE	s1, R	R := ((a1 + a2) / a3) + ((b1 – b2) / b3)

Leseaufträge: 3 MOVE Befehle, 5 anderes Befehle -> 3*1 + 5*2 = 13 Leseaufträge

Schreibaufträge: 8 Schreibaufträge

Rechenbefehle: 5

Berechnungszeit: (13+8) * 5 + 5 = 110 FLOP